FlatFeeStack
Fraud Resistant Multiplier Feature

Term Project

Autumn Term 2024

Version: 01.00
Date: 2024-12-19 20:43:44+01:00
Git Version: b4c4c8ac

Project Team: Yannick Staedeli
Mino Petrizzo
Roman Cvijanovic

Project Advisor/Leader: Dr. Thomas Bocek

Department of Computer Science
OST - University of Applied Sciences
Campus Rapperswil-Jona

Part I

Abstract

Abstract

Open-source software development relies heavily on community support and financial contributions. This Term Project presents a novel extension to the existing sponsorship platform FlatFeeStack. This project introduces two critical innovations: a *Multiplier Option* and a comprehensive *Fraud Prevention Mechanism*.

The *Multiplier Option* enables foundations to conditionally sponsor open-source repositories, wherein donations are triggered only after an initial user sponsorship. This approach establishes a collaborative funding model designed to encourage broader community participation. At the same time, it preserves the core values of FlatFeeStack: making the donation process as equitable, fair, and transparent as possible.

The *Fraud Prevention Mechanism* incorporates a sophisticated health scoring system that evaluates Git metrics and various internal sponsoring activities. Together, these six metrics form the Health Value. The scoring model enables administrators to review and evaluate repository assessments, allowing them to decide whether to certify the repository based on the provided evaluation. Moreover, only repositories within sponsoring bundles that include at least one verified repository are eligible to receive multiplier funding.

By integrating these features, the extension addresses key challenges in open-source funding, namely incentivize sponsorship and mitigating potential fraudulent activities. The implementation demonstrates a flexible, security-conscious approach to supporting open-source ecosystem sustainability.

Part II

Management Summary

Management Summary

Purpose

FlatFeeStack (view on GitHub) aims to transform open source development by creating a more sustainable and equitable compensation system. Working directly within the open source ecosystem, our project focuses on both supporting existing projects and developing new solutions to enhance developer compensation.

In our latest development phase, we implemented two key features to advance this mission. The Fee Multiplier introduces a system where Foundations provide additional support for repositories through dynamic bonuses, effectively multiplying the impact of user sponsorships. Complementing this, we developed the Repository Health Value system, which evaluates and identifies reliable repositories using objective metrics. This feature helps sponsors make informed decisions about which projects to support, increasing transparency and reliability in the open source funding ecosystem.

These implementations represent concrete steps toward our goal of creating a more sustainable open source environment where developers receive fair compensation for their contributions.

Key Achievements

Our development phase delivered two significant achievements that enhance the security and effectiveness of open source funding. The Fee Multiplier system implements an algorithmic approach to Foundation support distribution, incorporating fraud protection measures. This includes daily limits on support amounts and strict repository eligibility criteria, ensuring resources are directed to legitimate projects.

The Repository Health Value system introduces an evaluation framework based on six distinct metrics - two external and four internal measurements. These metrics combine to generate a score on a scale of 0 to 10, providing a clear assessment of repository quality. This scoring system serves as an additional layer of fraud protection by objectively measuring repository credibility.

A crucial innovation in both features is the implementation of FlatFeeStack's verify requirement. Repositories must either be verified by FlatFeeStack or be part of a sponsoring bundle including a verified repository before they can receive Foundation support. This verification process serves as a fundamental safeguard, ensuring the integrity of both the funding distribution and repository evaluation systems.

Our implementation addresses potential security vulnerabilities in the Fee Multiplier system. While FlatFeeStack enables both individual users and foundations to support projects, this flexibility could create opportunities for abuse. The initial concept of applying a straightforward multiplier to user

donations could have allowed malicious actors to manipulate the system and drain foundation funds as depicted in the figure below. To mitigate this risk, we implemented multiple safeguards in the Fee Multiplier system, ensuring the secure distribution of foundation support while maintaining the system's core functionality.

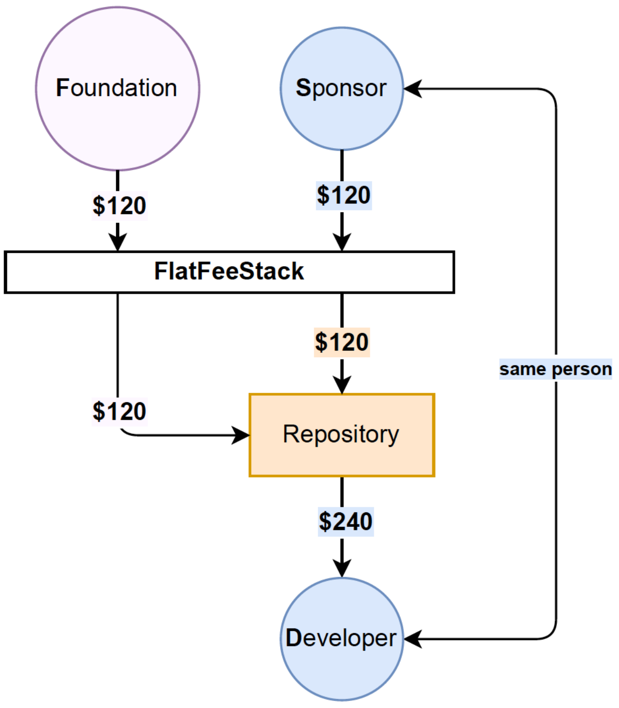


Figure 1: Developer exploiting weak Fee Multiplier feature

Figure 2 demonstrates the Fee Multiplier fund limitation and distribution: When a sponsor selects multiple repositories for contribution, the algorithm calculates a distribution pool based on the number of verified repositories. In the illustrated example, when at least one verified repository is present in the selection, the multiplier calculation proceeds. The pool is divided into six parts total, where each part represents a foundation's support for a repository. These parts are then distributed across the chosen repositories according to their foundation support levels, with repositories receiving multiple parts based on their level of foundation backing.

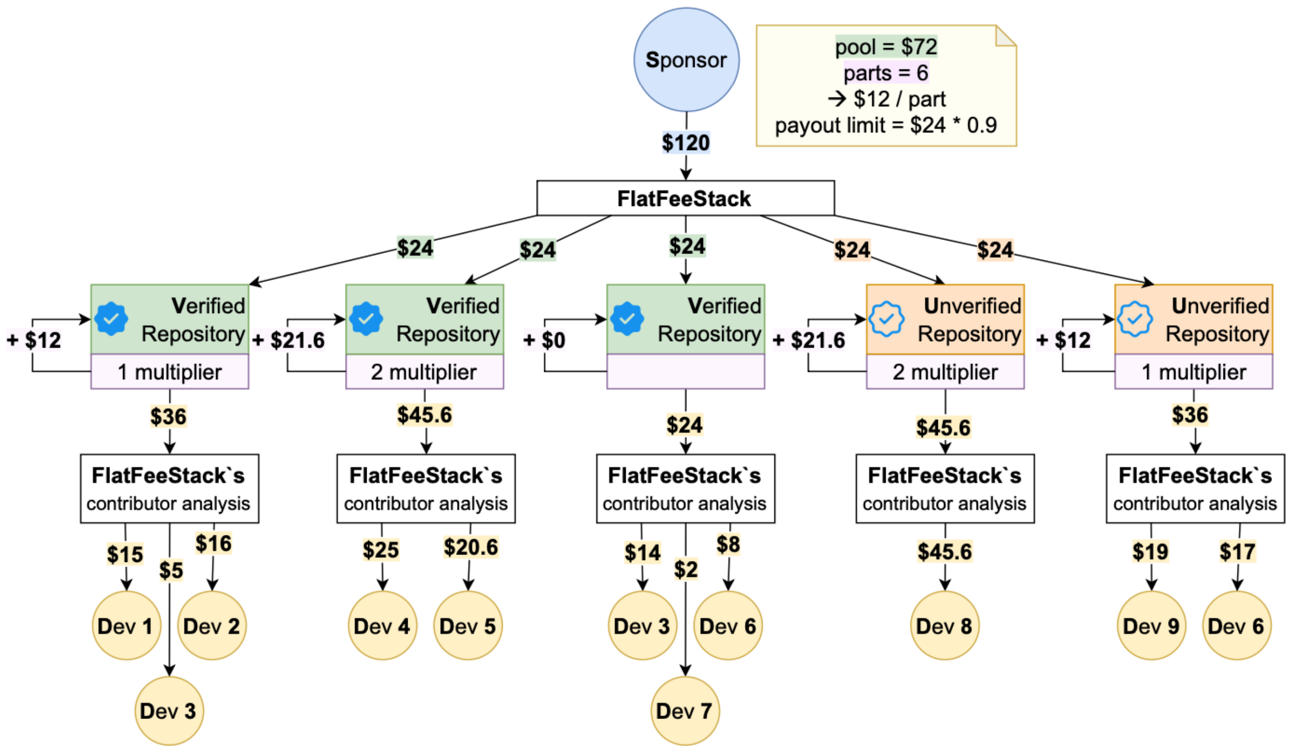


Figure 2: Key Achievement Example

A comprehensive list of our key implementations demonstrates how we enhanced user experience through interconnected features:

- Implementation of the Repository Health Value system, calculating scores based on six distinct metrics mapped against defined thresholds
- Development of a repository assessment dashboard that provides transparent visibility of all six metrics, offering detailed insights into each repository's status
- Integration of an administrative interface for verified repository selection, enabling manual verification of repositories
- Exposure of verified repository status to both foundations and users, increasing
- transparency in the ecosystem
- Implementation of the Fee Multiplier calculation system, enhancing repository support through algorithmic distribution of foundation funds

Timeline

This development was conducted as part of a Bachelor's degree term project in Information Technology, encompassing 720 hours of work between September 15 and December 20. The project allocated 16 hours per week and served dual purposes: deepening technical knowledge while gaining practical experience in structured agile development methodologies.

The work was organized into seven two-week sprints following scrum practices, with each sprint focusing on both functional and non-functional requirements. Each sprint concluded with concrete deliverables, ensuring steady progress toward our project goals. This structured approach allowed us to maintain consistent development velocity while systematically implementing and testing new features.

The scope and time allocation of the term project framework required careful management of the available 16 weekly hours, but we successfully delivered both major features - the Fee Multiplier and Repository Health Value system - within the semester timeframe.

Challenges

The project faced several significant technical and organizational challenges during implementation. Primary among these was integrating new features into an already complex code-base. Both the Fee Multiplier and Repository Health Value systems needed to work seamlessly with existing functionality, requiring careful consideration of system architecture and code dependencies.

A notable technical challenge was the team's initial unfamiliarity with Go. Despite being the project's primary programming language, none of the team members had extensive prior experience with it. This learning curve had to be managed alongside development responsibilities.

The project's scope presented additional complexity due to the interconnected nature of open source components. Understanding and maintaining these relationships while implementing new features required thorough system analysis and careful testing to prevent unintended consequences.

Time management proved challenging even with agile methodologies in place. The three-month timeframe, combined with the need for comprehensive planning and documentation, required constant prioritization and adjustment of development tasks to ensure timely delivery while maintaining code quality.

Contents

Part III

Glossary and Acronyms

Glossary

Active User An Active User is defined as a user who has donated to a Verified Repository within a specific time frame (currently three months). 48, 56

ActiveFFSUserCount The number of registered FlatFeeStack users who regularly donate to verified repos. 49, 57, 100, *see also* Health Value

CommitCount The cumulative amount of commits a repository has received in the last three months, indicating development activity. 49, 55, *see also* Health Value

ContributorCount Active contributors a repository has in the last three months, providing a measure of project activity and engagement. 48, 55, *see also* Health Value

Deduction Logic How does the amount, a single foundation has to pay, gets distributed among the contributors or also a single repo if future contribution. 61

Fee Multiplier A system that manages foundation contributions through a manually topped-up balance pool, from which daily deductions are made for sponsored repositories. ix, x, 50, 60, 98

Future Contributions When a repository receives stars and a payment is triggered, but has no registered developers on FlatFeeStack, it is saved in the future contributions table. Once a developer registers, they will receive the payment they are entitled to. 61

Health Value A calculated and built value describe the health of a given repo. 29, 48, 55

Metric Weight A coefficient assigned to each metric that determines its relative importance in the final Health Value calculation, with all weights summing to 10. 49, *see also* Health Value

Multiplier Sponsoring Money that is sponsored from a foundation or user through the multiplier feature. xiv, 3

Multiplier Sponsoring Limit The daily limit a foundation can set in the FFS Payment section, to restrict the maximum of Multiplier Sponsoring money they want to pay per day. 30, 43, 44, 46

MultiplierCount The number of active foundations that have set a multiplier for a specific repository. 49, 56, *see also* Health Value

PartialHealthValue Individual component value calculated from a single metric, its weight, and thresholds, which contributes to the overall Health Value. 48, 49, 55, *see also* Health Value

Payout Limit In a repository payout from a heterogeneous sponsoring bundle, the multiplier money is capped at 90% of the normal sponsoring amount. 30, 42, 61

Pool Calculation How to calculate the amount that can be distributed among the repos by the foundations. 61

repository A storage location where Git stores project files, along with their entire version history and configurations, enabling version control and collaboration. 27

SponsorCount The number of donations a project has received, indicating community trust and support. 49, 56, *see also* Health Value

Sponsoring Bundle The list which contains all repos a user has marked with a star. Meaning, all repos the user donates to. 3, 30

StarCount The total number of stars a repository has received on FlatFeeStack from users who sponsor at least one verified repo. 49, 56, 100, *see also* Health Value

Threshold A lower and upper bound used to linearly map raw metric values in the Health Value calculation, where both bounds are inclusive. 50, 57, *see also* Health Value

Unverified Repository A repository that FlatFeeStack has assessed with 2 external and 4 internal metrics, and afterwards marked as unverified. OR, a repository that has not been assessed yet. 29

Verified Repository A repository that FlatFeeStack has assessed with 2 external and 4 internal metrics, and afterwards marked as verified. xiv, 29

Acronyms

AGB	Allgemein Geschäftsbedingungen (German for: Terms and Conditions). 38
FFS	FlatFeeStack. 15, 29, 49, 60, 104
NFR	Non Functional Requirement. 101

Part IV

Product Documentation

Chapter 1

Introduction

This report details the development of the Fraud Resistant Multiplier Feature in the FlatFeeStack software, a Term Project at the Eastern Switzerland University of Applied Science (OST).

The aim of this chapter is to explain what parts of FlatFeeStack already existed and which features were developed during the semester. Within this chapter we also present our motivation why we have chosen this project, and the general conditions.

1.1 Existing Project

Since we are working on an existing project, it's essential to explain what this application is already capable of and its intended purpose. The project, FlatFeeStack, is fundamentally designed to financially support open-source developers.

While other donation services focus on supporting individual projects or developers, The core principle of FlatFeeStack is to make the process as evenly, fair and transparent as possible to encourage donations. For $120 per year, everyone can support any open-source project, with the donation evenly distributed among the projects and active developers. For companies, this is easier to budget as it comes down to a flat fee per developer. The distribution of funds to open-source contributors is based on how much code the developer has changed in the last three months.

If someone wishes to support an open-source project through FlatFeeStack, they follow these steps:

1. Create an account on FlatFeeStack.
2. Select the repositories they wish to support — these are listed within the program, and users can search for them.
3. Choose a payment method: monthly, yearly, or a five-year plan.
4. The funds are fairly distributed among the selected repositories.

Each of these steps will be explained in more detail as we proceed with the work.

To get a great overview of the existing features and technical implementations, take a look at this list:

- User authentication
- analyzer of git repositories (for sponsoring distribution only)
- reverse proxy setup (caddy)
- database setup and each entry needed for existing features. The edited and newly created tables are clearly marked as such in our C3 model.
- Each step needed to make a "normal" sponsoring (frontend and backend)
  - Connection of payment methods (using Stripe)
  - Search through all existing git projects
  - add git projects to a list that will receive your donations
- connection with git account for payout service
- daily payout of summarized donations for a developer
- everything around DAO
- everything around crypto currency
- bylaws
- monitoring

1.2 Assignment

FlatFeeStack is a fully functional web application, currently not yet publicly deployed. Before its official launch, our advisor, Thomas Bocek, proposed the addition of a new feature.

The main goal of this additional feature, is the implementation of a fraud resistant multiplier. The multiplier can be assigned by a foundation or also a normal FlatFeeStack user to a repository. The initial idea of the feature was, that each time a repository receives a donation, the one who assigned the multiplier will pay the same amount on top. To break this process down: (1) Foundation / User assigns the multiplier to a git project XY (2) Another user which added this git project XY to his Sponsoring Bundle pays $20 per day and $5 goes to Project XY. (3) Project XY receives $5 from user sponsoring + $5 from foundation Multiplier Sponsoring.

When a developer realizes they can receive double the amount they pay, it creates a strong incentive to exploit the system. Consequently, addressing fraud concerns necessitated the implementation of robust fraud protection measures.

So, establishing a plan to make the new multiplier feature as secure as possible is the second main goal and also a huge part of this term project. Developing an appropriate solution required a deeper understanding of security analysis. Utilizing this expertise, a comprehensive risk analysis was conducted. Additionally, reinventing the multiplier is the only viable option, which is expected to result in a far less risky donation feature designed for foundations.

The primary objective is to build a secure application that earns the recognition of foundations and users, motivating them to utilize the platform.

1.3 Motivation

Our motivation for participating in this project primarily revolves around gaining and deepening our expertise with the technology stack. FlatFeeStack follows a classic Client/Server architecture. The presentation layer is built with Svelte, while both the business logic and data access layers are implemented in Go. Communication between these three layers is handled through HTTP methods.

By contributing to FlatFeeStack, we expect to not only gain considerable expertise in both Svelte and Go but also develop a deeper understanding of how these components interact and work together. The interconnected nature of these technologies provides an excellent opportunity to expand our knowledge of modern web architecture.

1.4 General Conditions

This Computer Science Term Project is developed during the Autumn Term of 2024. Our team consists of three members, giving us a total time budget of 720 hours. The project earns 8 ECTS credits, with our university estimating 30 hours of work per ECTS. This amounts to 240 hours per person, or approximately 17 hours per week.

The exact time spent on tasks is tracked using the app "Timetracker" as an extension of Jira. The final time reports are available at the end of this documentation under Timer Reports.

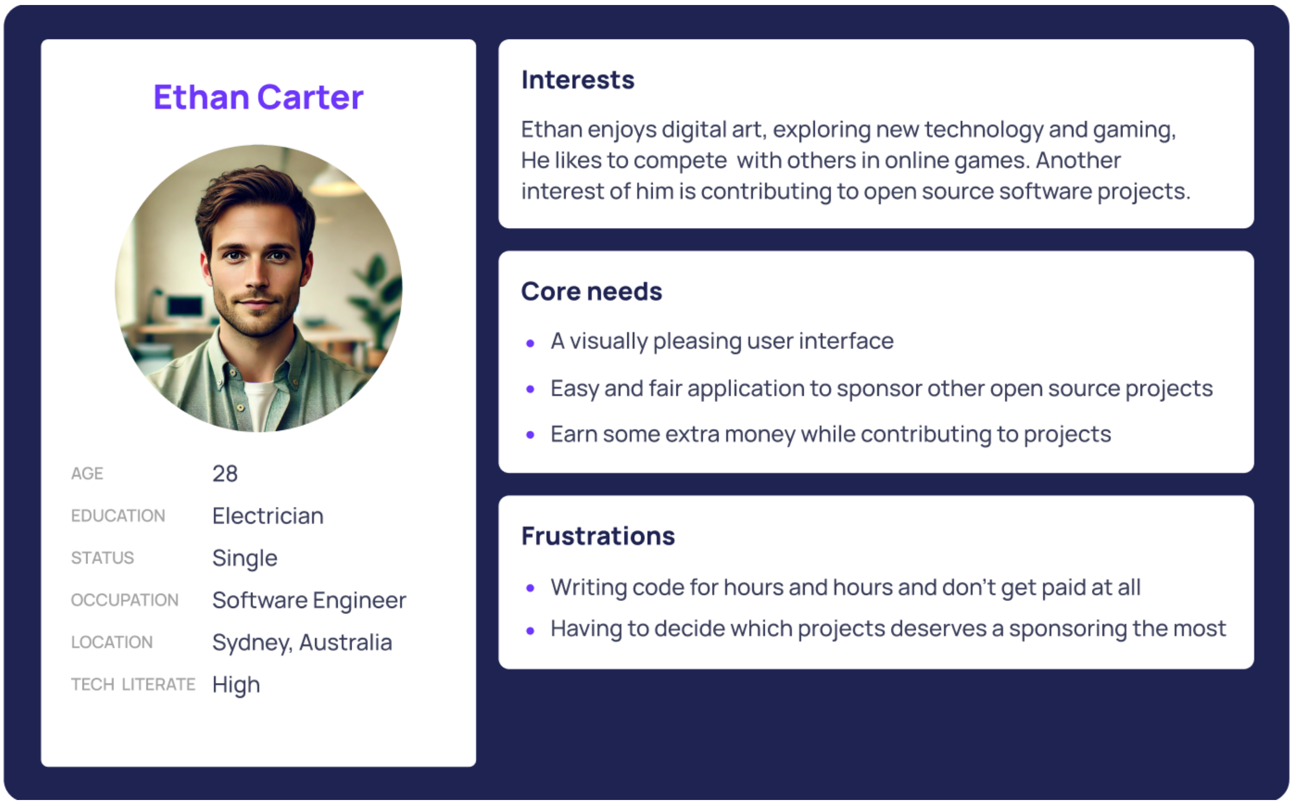
Chapter 2

Requirements

This chapter defines the system requirements, starting with Personas and Actors to identify key users and interactions. A Use Case Diagram provides an overview, followed by detailed Functional Requirements for system capabilities like selecting projects and managing multipliers. Finally, Non-Functional Requirements ensure the system meets standards for accuracy, performance, and security, forming a solid foundation for development.

2.1 Personas

We created two personas with differing and distinct characteristics. We believe them to be typical users of our application.



Ethan Carter

AGE	28
EDUCATION	Electrician
STATUS	Single
OCCUPATION	Software Engineer
LOCATION	Sydney, Australia
TECH LITERATE	High

Interests

Ethan enjoys digital art, exploring new technology and gaming, He likes to compete with others in online games. Another interest of him is contributing to open source software projects.

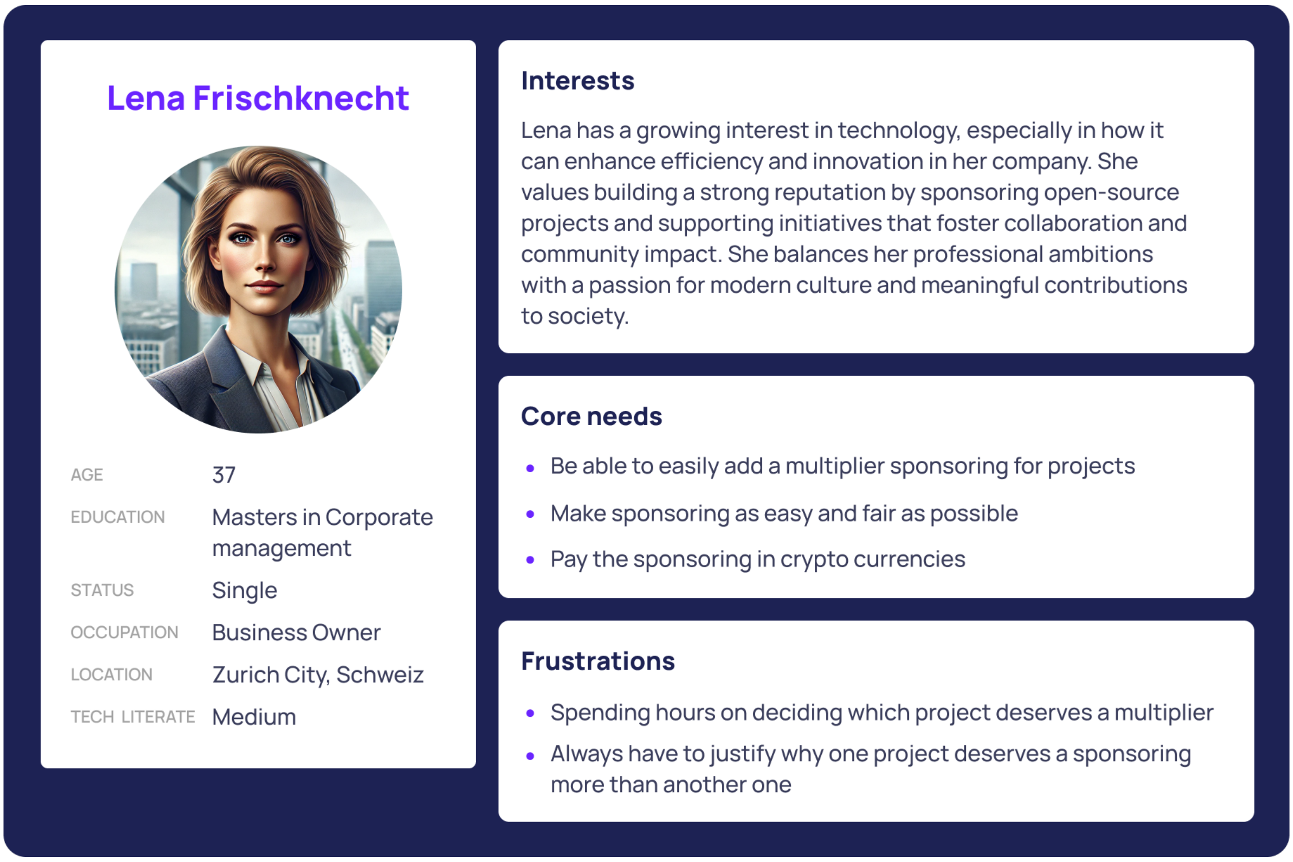
Core needs

- A visually pleasing user interface
- Easy and fair application to sponsor other open source projects
- Earn some extra money while contributing to projects

Frustrations

- Writing code for hours and hours and don't get paid at all
- Having to decide which projects deserves a sponsoring the most

Figure 2.1: Persona User



Lena Frischknecht

AGE	37
EDUCATION	Masters in Corporate management
STATUS	Single
OCCUPATION	Business Owner
LOCATION	Zurich City, Schweiz
TECH LITERATE	Medium

Interests

Lena has a growing interest in technology, especially in how it can enhance efficiency and innovation in her company. She values building a strong reputation by sponsoring open-source projects and supporting initiatives that foster collaboration and community impact. She balances her professional ambitions with a passion for modern culture and meaningful contributions to society.

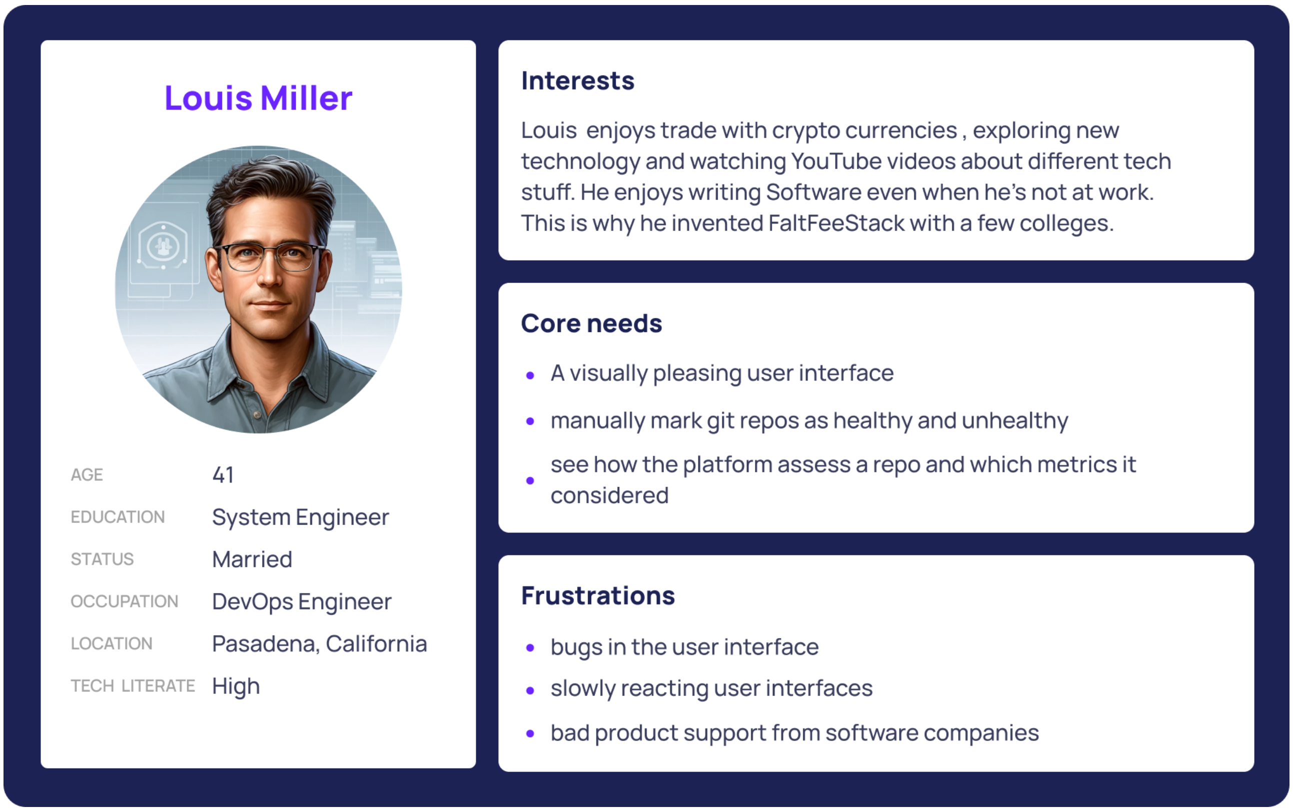
Core needs

- Be able to easily add a multiplier sponsoring for projects
- Make sponsoring as easy and fair as possible
- Pay the sponsoring in crypto currencies

Frustrations

- Spending hours on deciding which project deserves a multiplier
- Always have to justify why one project deserves a sponsoring more than another one

Figure 2.2: Persona Foundation



Louis Miller

AGE	41
EDUCATION	System Engineer
STATUS	Married
OCCUPATION	DevOps Engineer
LOCATION	Pasadena, California
TECH LITERATE	High

Interests

Louis enjoys trade with crypto currencies , exploring new technology and watching YouTube videos about different tech stuff. He enjoys writing Software even when he’s not at work. This is why he invented FaltFeeStack with a few colleges.

Core needs

- A visually pleasing user interface
- manually mark git repos as healthy and unhealthy
- see how the platform assess a repo and which metrics it considered

Frustrations

- bugs in the user interface
- slowly reacting user interfaces
- bad product support from software companies

Figure 2.3: Persona Admin

These personas will help us to better define the requirements of our application, while taking in regards the fairly limited time frame for the scope of this project. Nevertheless we are determined to implement the main feature, Fee Multiplier, in the given time frame. We strife to implement it such, that the sponsoring feels clear and safe.

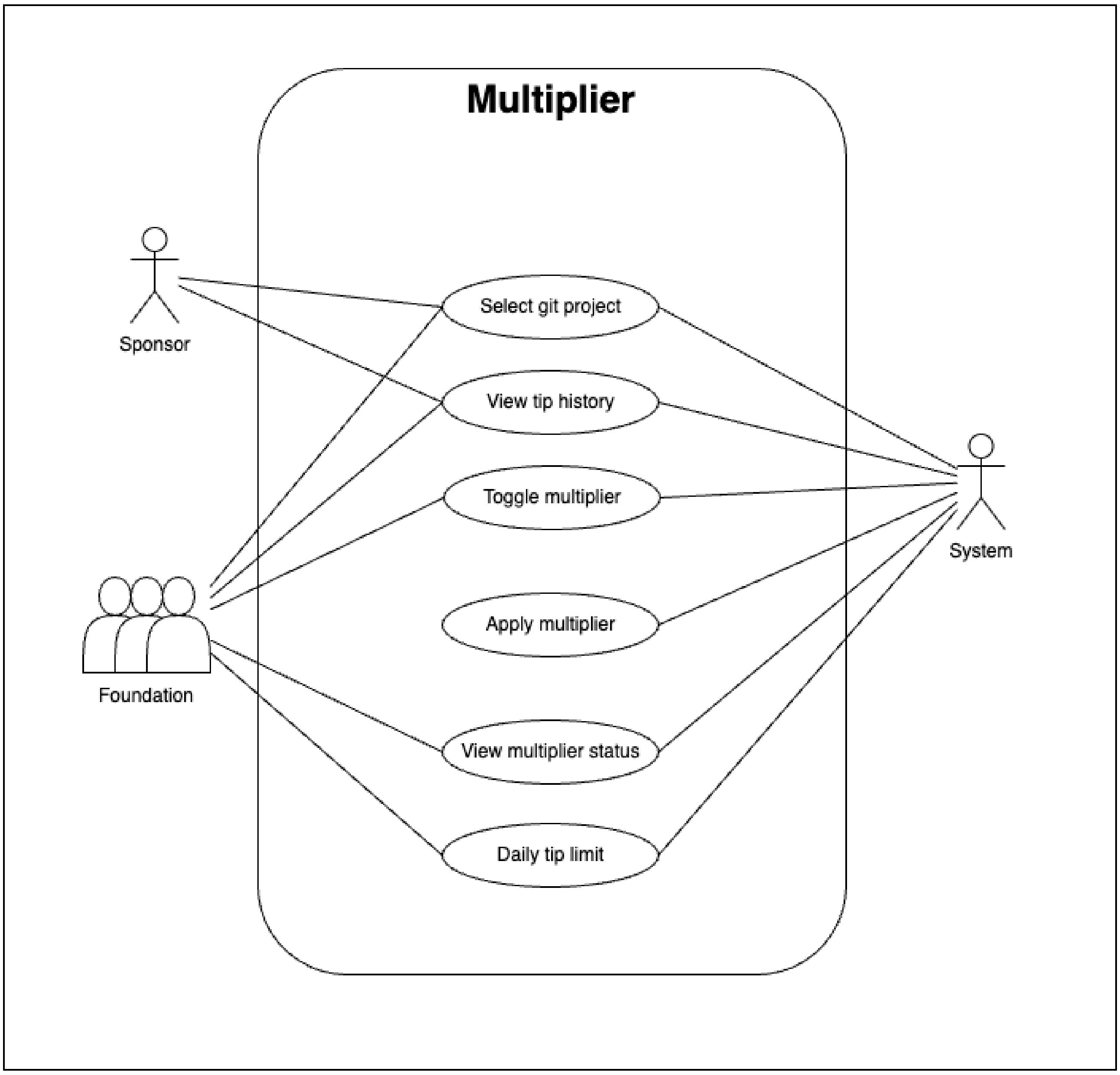
We also created a persona representing what we envision s a typical administrator. This step is crucial because our scope primarily focuses on implementing the admin console features of the existing application. Additionally, the console provides valuable insights into repositories, ensuring that administrators can access and oversee them efficiently.

The persona cards were designed using Figma, and the profile pictures were generated with ChatGPT. [Ope24] [Fig24]

2.2 Actors

- **Sponsor**: A person or foundation providing sponsoring (monetary) to Git repository.
- **Foundation**: A person or foundation enabling additional support to Git repository by applying the fee multiplier.
- **System**: The system that handles tipping, multiplier management, and related actions.

2.3 Use Case Diagram



2.4 Functional Requirements

This section details the functional requirements of the system, based on the use cases outlined in the diagram.

2.4.1 FR1: Select Git Project

Main Success Scenario

- A sponsor or foundation member selects a Git repository from the available list of projects or provides the Git repo link.

- The system displays the details of the selected Git repository like current multiplier status (only for foundation), multiplier count (for all users), and other related information.

Alternate Scenarios

- If the selected Git project is not available or has been removed, nothing gets displayed.
- If the user is not yet registered as a valid sponsor or foundation member, they are redirected to the registration page.

2.4.2 FR2: View Multiplier Sponsoring History

Main Success Scenario

- The user (sponsor or foundation) can view a history of all Multiplier Sponsoring given to a specific Git repository.
- The system presents a list of multiplier sponsoring transactions, including amounts, dates, and whether a multiplier was applied (everything anonymously).

2.4.3 FR3: Toggle Multiplier

Main Success Scenario

- A foundation member enables or disables the multiplier for a specific Git repository.
- If activated, the multiplier (e.g., 2x) will apply to future sponsoring made to the repository.
- The system confirms the state (on/off) of the multiplier.

2.4.4 FR4: Apply Multiplier

Main Success Scenario

- When a sponsor makes a donation to a Git project, the system checks if a multiplier is active on a verified repo.
- If a multiplier is active (e.g., 2x), the system automatically applies it, multiplying the amount of contribution made by the sponsor.
- Both the sponsor and the foundation contribution are logged in the payment history.

Alternate Scenarios

- If no multiplier is active, the system processes only the sponsoring and updates the payment history.

- If the foundation's daily multiplier sponsoring limit (see FR6) has been exceeded, the foundation gets excluded from the multiplier calculation.
- If there are no verified repos inside the sponsors bundle, the multiplier is not applied

2.4.5 FR5: View Multiplier Status

Main Success Scenario

- A user (sponsor or foundation) views the current multiplier status on a Git repository.
- The system displays the active multiplier (if any) and the amount of foundation support.

Alternate Scenarios

- If no foundation has the multiplier applied on that specific repo, the system displays 0x for that repo.

2.4.6 FR6: Daily Multiplier Sponsoring Limit

Main Success Scenario

- A foundation can view and set a daily multiplier sponsoring limit as the total amount it can contribute via multipliers across all Git repositories with active multiplier each day.
- During the daily contribution check, the system validates if the daily multiplier sponsoring limit has been exceeded.
- If the limit is not exceeded, the multiplier is applied.

Alternate Scenarios

- If the foundation's daily multiplier sponsoring limit is close to being exceeded (e.g., the remaining balance is insufficient to match the full multiplier on a sponsoring), the system applies only the portion of the multiplier that fits within the remaining balance. For example, if the foundation has $50 left for the day and a sponsor gives a $100 tip with a 2x multiplier, the system will only match the sponsor's tip up to the remaining $50.

2.5 Non-Functional Requirements

2.5.1 NFR1: Multiplier Accuracy

Category: Accuracy $\rightarrow$ Computational Precision
Description: The multiplier functionality must calculate sponsorings with a precision of microcents of six decimal places, ensuring minimal rounding errors occur during the application of the multiplier

to a sponsoring.
Acceptance Criteria: The system must correctly apply the multiplier to any sponsoring amount, producing a value with a maximum deviation of ± 0.01 from the expected result.
Verification Process:

- Apply different multiplier values to a set of predefined sponsoring amounts.
- Compare the calculated results to the expected values.
- Ensure the results are consistently accurate to two decimal places.

Verification Period: During testing and before each major release involving changes to the sponsoring or multiplier functionality.

2.5.2 NFR2: Performance under Concurrent Use

Category: Performance $\rightarrow$ System Load
Description: The multiplier extension must handle up to 1000 concurrent users without affecting the response time of the existing tipping system. The response time must remain within 500ms, even with the added calculations from the multiplier.
Acceptance Criteria: The system's performance should not degrade by more than 5% after implementing the multiplier extension, maintaining tip processing speeds at under 500ms for 95% of requests under a concurrent user load of 1000.
Verification Process:

- Conduct load testing with and without the multiplier extension activated, simulating up to 1000 concurrent users.
- Measure the response times for tip calculation requests to assess any impact on performance.
- Ensure performance remains within the acceptable range under standard and peak loads.
- Perform tests using the hardware specifications of a Raspberry Pi 5 to simulate real-world performance constraints.

Verification Period: Continuous testing during load tests and performance reviews after integration.

2.5.3 NFR3: Dynamic Multiplier Calculation

Category: Functionality $\rightarrow$ Dynamic Configuration
Description: The system must automatically calculate and apply a multiplier based on the devised algorithm, with a crucial constraint tied to repo health. Admins have the capability to designate a repo as verified, and the system requires at least one such repo within a user's repo bundle to activate and enable the multiplier for payout. The calculation logic must be flexible while simultaneously ensuring that the sponsoring does not exceed the imposed payout limit.

Acceptance Criteria: The multiplier applied to any sponsoring must be below the imposed payout limit. The system should dynamically adjust the multiplier based on selected repositories and available funds. This approach serves as a fraud protection mechanism, designed to minimize the risk of unauthorized or malicious payouts by implementing stringent health and verification criteria.
Verification Process:

- Test the system under various sponsoring conditions to trigger the multiplier calculation.
- Verify that the calculated multiplier is below the maximum allowed sponsoring for a given repo per foundation sponsoring.
- Confirm that the multiplier activation is strictly dependent on having at least one verified repo in the user's repo bundle.
- Simulate edge cases to ensure the multiplier does not exceed established boundaries.

Verification Period: Regular testing during system operations, with comprehensive verification in both staging and production environments before major releases.

2.5.4 NFR4: Secure Multiplier Access

Category: Security $\rightarrow$ Access Control
Description: Only authorized system components should have the ability to calculate the multiplier. The multiplier logic should be protected against unauthorized access or manipulation. Any interaction with the multiplier must be authenticated and logged for auditing purposes.
Acceptance Criteria: The system must authenticate all requests that attempt to calculate or update the multiplier. Any unauthorized attempts to access or manipulate the multiplier should be blocked and logged with details like the source IP address and user identifier.
Verification Process:

- Ensure that only authenticated system services can initiate multiplier calculations.
- Simulate unauthorized access attempts and confirm that they are denied and logged.
- Review the logs to ensure unauthorized access attempts are properly recorded with all necessary details.

Verification Period: Security tests in every sprint, with detailed audit log reviews at the end of each release cycle.

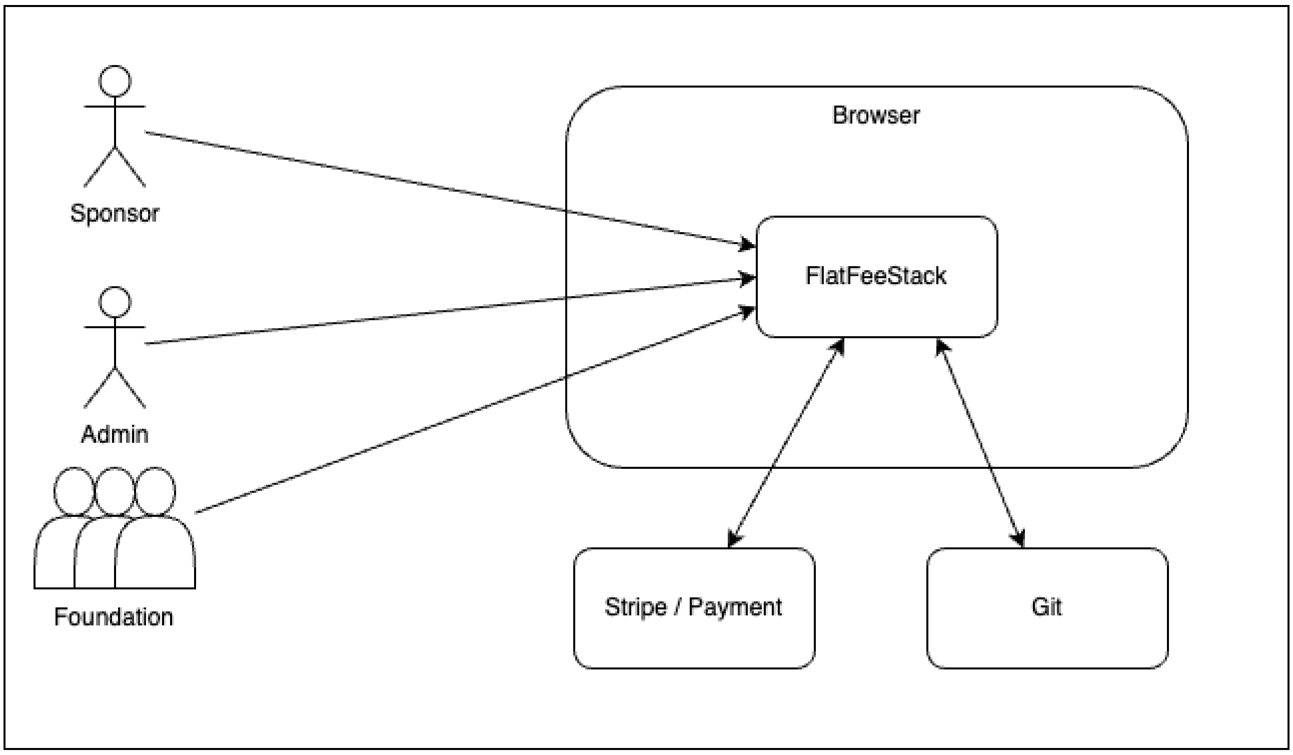
Chapter 3

Architecture

This chapter presents the system architecture, summarizing the System Context Diagram (C1), Web Architecture (C2), and Component Overview (C3) to define the system's scope, design, and key components. It also provides details about the Frontend, Backend, and Analyzer, including their technologies and folder structures, along with the Database and Client-Server Cut (CSC) to explain data flow and cross-cutting concerns.

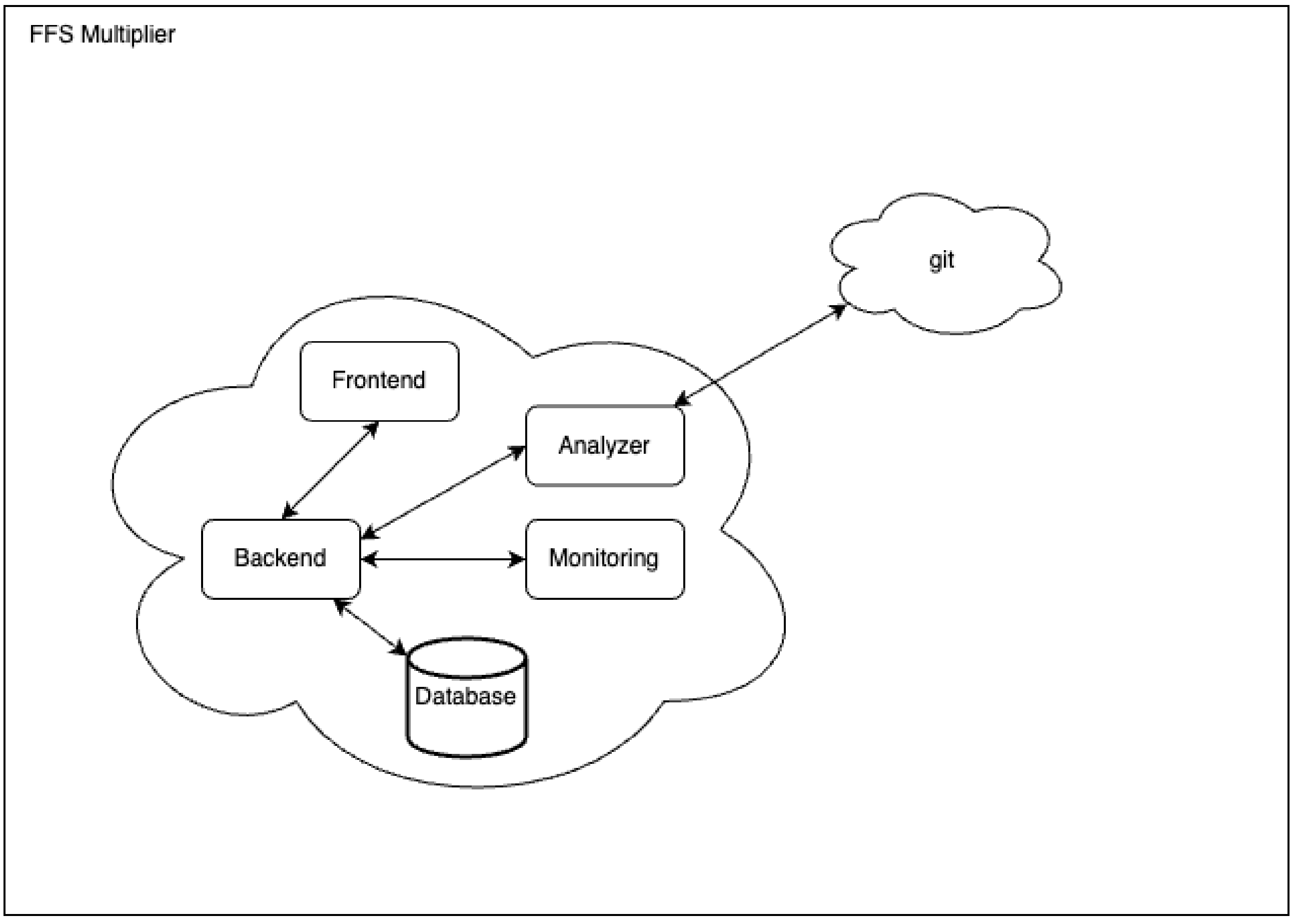
3.1 System Context diagram (C1)

The System Context Diagram outlines the system's interactions with external entities, providing a high-level view of its scope and relationships.



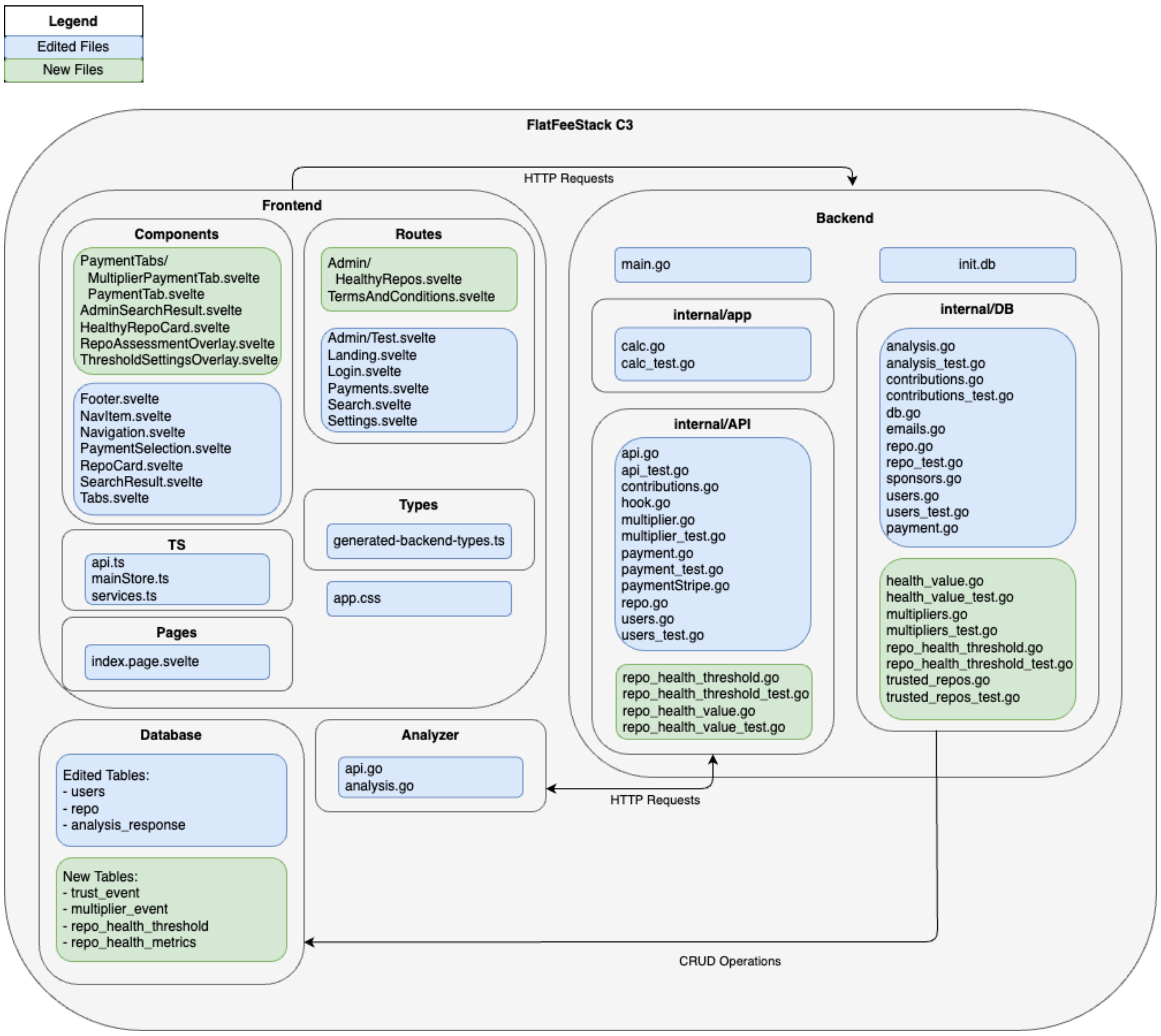
3.2 Web Architecture (C2)

The Web Architecture explains the structural organization of the system's web components, focusing on frontend-backend communication.



3.3 Component Overview (C3)

The Component Overview breaks down the system into its main components, highlighting their roles and integration within the architecture. This approach of C3 model specifically outlines the difference between edited and newly created files.



3.4 Decisions

As we are working on an existing project, the file structure and the programming languages are given. This means that we will make changes in the following folders, which are described below: frontend, backend, analyzer, database and monitoring. The programming language we use in the backend is Go, which is a simple, efficient and compiled programming language. In the frontend we will use the library Svelte, where we will use HTML, CSS and TypeScript.

3.5 Frontend

In the Frontend subsection, the technologies used in FFS are introduced, along with an explanation of the folder structure. This aims to clarify the purpose of key files and directories, making it easier for new developers to navigate and understand the project.

3.5.1 Technologies Used

The frontend utilizes several web development technologies (already in use):

- **Svelte**: A modern JavaScript framework for building user interfaces. Unlike frameworks like Vue.js and React, which update the DOM using a virtual DOM, Svelte compiles components into highly efficient JavaScript during build time, producing minimal and optimized code.
- **TypeScript**: A statically typed language that compiles down to JavaScript, helping to catch errors early during development.
- **Prettify**: Automatically format code for consistency and readability.
- **Vite**: A build tool that provides fast server startup and hot module replacement for efficient development and production builds.
- **Caddy**: A web server for serving the application, configured through the `Caddyfile`.
- **PNPM**: A package manager, similar to npm or yarn, which is used to install and manage dependencies.
- **Docker**: Containers are defined using Dockerfiles for consistent environments during development and production.

3.5.2 Folder Structure

The frontend is well-organized with clear separation of concerns. Below is a breakdown of the most important parts of the structure:

- `Caddyfile`: Configuration file for Caddy, used for serving the frontend.
- `Dockerfile` and `Dockerfile.dev`: Docker configuration files for production and development environments respectively.
- `index.html`: The main HTML file where the Svelte application is mounted.
- `public/`: Directory containing static assets such as images, icons, and manifest files.
  - `images/`: Contains various SVG and image assets used in the frontend.
- `src/`: The source code of the application.
  - `components/`: Contains reusable UI components built with Svelte, which are then used on different `routes/`.
  - `contracts/`: TypeScript files that handle interactions with smart contracts.
  - `pages/`: Different pages for routing, including landing pages, admin pages, and DAO-related views.

    * `index.page.svelte`: Contains all views of the website, either only accessible for authenticated users (`<PrivateRoute>`) or public (`<Route>`).
  - `routes/`: Svelte files that define different routes for the application.
  - `ts/`: Contains TypeScript files for state management, services, and utilities.
    * `mainStore.ts`: Contains all reactive store-variables used for the multiplier feature and admin console, enabling state updates and synchronization across different components.
    * `api.ts`: Contains all API's, connecting the frontend to the backend.
  - `types/`: Contains all custom types, used to correctly program components in typescript.
    * `backend.ts`: Contains all the Backend types, that can be used in TypeScript.
    * `generated-backend-types.ts`: Can be generated with the command *'npx openapi-typescript ../backend/backend.yaml –output ./src/types/generated-backend-types.ts'*. This command is also part of the `package.json`.
  - `utils/`: Utility functions such as Ethereum address formatting and validation helpers.

- `package.json`: Defines the project's metadata, dependencies, scripts, and configuration for Node.js applications.
  - `Important Scripts`: *'prettify'* to auto format clean code / *'schema:backend'* to auto generate the backend types.

- `tsconfig.json` and `tsconfig.node.json`: TypeScript configuration files defining compiler options.

- `vite.config.ts`: Vite configuration file for bundling and development setup.

- `svelte.config.js`: Configuration file for Svelte, defining how components are compiled.

3.6 Backend

Similar to the Frontend subsection, the Backend subsection provides an overview of the key files and directories, explaining their purpose and role within the project.

3.6.1 Technologies Used

The backend employs several technologies for web development and microservices:

- **Go (Golang)**: A statically typed, compiled programming language known for its simplicity and performance, especially in concurrent and web applications.
- **Docker**: Used to create containerized environments for both development and production, defined by `Dockerfile`.
- **YAML**: The configuration is managed using a `backend.yaml` file, which holds settings and parameters in a human-readable format. `backend.yaml` employs the Open API Specification.
- **SQL**: The database interactions are handled through SQL files, such as `init.sql`, which define schema creation and initial data population. For compatibility and testing purposes the sql syntax is simplified and works with PostgreSQL and SQLite.
- **Go Modules**: Dependencies are managed using Go's native module system, with `go.mod` and `go.sum` files to lock dependencies and ensure consistent builds.

3.6.2 Folder Structure

The backend folder structure is organized for scalability and maintainability. Below is an overview of the key components:

- `Dockerfile`: Defines the instructions for building the Docker image for the backend.
- `backend.yaml`: YAML configuration file for the backend service.
- `db/`: Directory for SQL files that handle database initialization and data setup.
  - `init.sql`: SQL script to initialize the database schema.
- `internal/`: This is where the main business logic is stored, organized into sub-packages:
  - `api/`: Contains the API layer, including endpoints for handling contributions, payments, users, email hooks and task delegation to business logic layer. `api/` acts like a controller.
  - `app/`: Contains application-level logic, such as calculations and domain-specific operations. This is purely business logic which is mainly executed through automated tasks.
  - `client/`: Handles integration with external services like GitHub and email services.
  - `cron/`: Contains cron jobs for scheduled tasks and background operations.

  - `db/`: Defines how the backend interacts with the database, with modules for handling analysis, contributions, payments, and user management. This layer purely describes data access.
  - `handler/`: Contains request handlers, such as those managing user-related logic.
- `main.go`: The entry point for the Go backend application, where the server is initialized and run.
- `pkg/`: Contains reusable packages:
  - `config/`: Configuration management.
  - `middleware/`: Middleware components, including JWT handling and logging.
  - `util/`: Utility functions for testing, time handling, and other general-purpose helpers.
- `templates/`: Directory for email templates used for sending notifications to users.
  - `email/en/`: Email templates in both HTML and plain text formats, categorized by their purpose (e.g., payment notifications, marketing).

3.7 Analyzer

The Analyzer, a critical component of our product, operates as a separate Docker instance. Written in Golang, it functions as an extension of the backend. Notably, the backend and the Analyzer instance communicate via HTTP.

3.7.1 Technologies Used

The analyzer utilizes a range of technologies to perform its function efficiently:

- **Go (Golang)**: Same as in backend
- **Docker**: Same as in backend
- **HTTP API**: The analyzer exposes its functionality via HTTP endpoints, defined in `api.go` and testable through `api.http`.
- **Go Modules**: Same as in backend

3.7.2 Folder Structure

The folder structure of the analyzer is simple yet well-organized, making it easy to navigate and extend:

- `Dockerfile`: Specifies the instructions for building a Docker image to run the analyzer service.
- `analysis.go`: The core of the analyzer logic, which performs the main analysis functions.
- `analysis_test.go`: Unit tests for the analysis logic to ensure its correctness and reliability.

- `api.go`: Defines the HTTP API endpoints that expose the analyzer's functionality to external clients or services.
- `api.http`: A test file containing HTTP requests that can be used to test the API manually or through automated testing tools.
- `banner.txt`: A text file that contains a banner or welcome message displayed when the analyzer service starts.
- `go.mod`: Specifies the Go module path and lists the dependencies required by the project.
- `go.sum`: A file that locks the dependency versions to ensure consistent builds across different environments.
- `main.go`: The entry point of the Go application, responsible for initializing the service and running the main logic.
- `test-repository.zip`: A test archive that contains sample data for testing the analyzer's functionality.

3.8 Database

Another part of the backend is the database. In use is PostgreSQL. For the initialization of the database docker an `.env` is used. The keep the configuration simple only very few tweaks were performed.

3.9 Client-Server Cut (CSC)

The Client-Server Architecture consists of a set of layers on both the client (frontend) and server (backend) sides. Each layer has specific responsibilities, allowing for a clear separation of concerns and maintainable code.

3.9.1 Client (Frontend)

The frontend (client side) is responsible for user interaction and sending/receiving requests to/from the backend. It is structured into the following layers:

Presentation Layer

- **Responsibility:** Handles the user interface (UI).
- **Components:**
  - UI Components (Svelte Components, HTML, CSS)
  - Presentation Logic (handling user inputs)

Service Layer

- **Responsibility:** Acts as a bridge between the frontend and backend by handling API requests and responses.
- **Components:**
  - HTTP Client
  - Error Handling for API calls

3.9.2 Server (Backend)

The backend (server side) is responsible for processing client requests, business logic, and managing the data layer. The server is structured into the following layers:

Presentation Layer (API Endpoints)

- **Responsibility:** Exposes API endpoints that the client can interact with.
- **Components:**
  - RESTful API Endpoints

Service Layer

- **Responsibility:** Implements the core business logic and processes client requests.
- **Components:**
  - Business Logic Services (e.g., Payment Services, User Services)
  - Validation of client requests

Business Layer

- **Responsibility:** Coordinates workflows and encapsulates complex business rules.
- **Components:**
  - Application Facades (to streamline interaction with the service layer)
  - Business Components (individual business functions)
  - Business Entities (data models representing domain concepts)

Data Layer

- **Responsibility:** Manages data storage, retrieval, and database interactions.
- **Components:**
  - Data Access Components (SQL queries, ORMs, repositories)
  - Data Helpers & Utilities (data validation, formatting)
  - Service Agents (interfaces to external services or APIs)

3.9.3 Cross-Cutting Concerns

These aspects affect multiple layers of the system, both on the client and server sides. They ensure proper security, monitoring, and reliability of the entire architecture.

Security

- Ensures that both client and server layers handle authentication, authorization, and data encryption.

Logging

- Responsible for recording system behavior, especially errors and important events.

Error Handling

- Handles exceptions and provides feedback in case of system or application errors.

3.9.4 Data Flow

- The client sends requests to the server through the **Service Layer**, which is then processed by the backend's **Service** and **Business Layers**.
- The **Data Layer** on the server side retrieves or stores data in the data source (database, external services).
- Results or error messages are sent back to the client, where the **Presentation Layer** updates the UI.

Chapter 4

Domain Analysis

For the development of the new feature, we conducted an in-depth exploration of security auditing, as one of the primary objectives is to create a fraud-resistant solution.

4.1 Security Auditing

In this section we provide some background knowledge we needed, to start with identification of threads.

> *Security auditing is a systematic process of examining a system, application, or network to ensure it adheres to security policies, best practices, and regulations. It aims to identify vulnerabilities, potential threats, and weaknesses that could be exploited by attackers. To ensure financial fraud is one of the main goals of a security audit. [ISA22]*

Security is essential for our Project, because the quality of our project rises and falls with the security quality. Security auditing covers various approaches, we will utilize some of them to improve our solution with.

To execute this task as professionally as possible, we held a brief meeting with our Cyber Security and Cyber Defense lecturer, Ivan Bütler. He provided valuable insights into best practices used in his company. Two of the security auditing methodologies he shared have proven particularly useful to us.

- **Threat Modeling:** This involves identifying the potential risks associated with a feature or component in a project, estimating the likelihood of these risks occurring, and assessing their potential impact. In our case, this would involve analyzing how the multiplier could be manipulated for fraud and what damage that could cause. [Fou24] [Bue24]
- **Risk Analysis:** In a business context, you would present the findings from the Threat Modeling process to the customer, discussing which threats they would like to prioritize for further attention. The security company would then conduct deeper investigations into these selected threats through a process called Risk Analysis. This Risk Analysis provides a more detailed assessment, focusing on the likelihood and impact of each threat, going beyond the initial Threat Modeling. [Dev24] [Bue24]

Based on input from Ivan Bütler, we have decided to document only the Risk Analysis. As product owners, together with our advisor, we will determine our own priorities and decide which areas to focus on.

Chapter 5

Solution Strategy

As previously discussed, implementing a multiplier that is entirely secure against fraud is not feasible. To mitigate the risk of fraud, we have to create a solution, where it is as difficult as possible to violate our rules. This requires adopting the perspective of an adversary attempting to exploit the system and steal money. This chapter aims to identify as many vulnerabilities as possible which could lead to misuse of FlatFeeStack, and creating a solution strategy to prevent such occurrences.

5.1 Identification of Fraud Techniques

In this section, provided is an overview of how an adversary might attempt to steal money through the platform. Beginning by conducting a risk assessment, identifying potential methods of theft, followed by risk mitigation strategies. Each topic is presented in a table format, where each identified risk is assigned a reference number, which is used in the mitigation table for cross-referencing.

If our mitigation strategy reveals additional vulnerabilities, we assign updated risk numbers accordingly. The following section (Fraud Prevention Techniques) details our mitigation strategies, including an analysis of techniques employed to prevent system abuse. We begin by outlining key tasks for implementing a robust fee multiplier system. This is followed by a discussion of planned features that will enhance decision-making for git repository support.

All the visuals in this section were created using the online tool diagrams.net (formerly draw.io). [LdA24]

5.1.1 Risk Assessment

To identify the potential risks of our multiplier feature we utilize the previously explained Risk Analysis.

Risk Number:
- Risk 1-9: fundamental risks with multiplier feature
- Risk 10-19: followup risks from mitigation strategies of fundamental risks
- Risk 20-29: general risks with FFS (may already been solved)

Risk Score Calculation:
- L represent the **Likelihood** score, where:

$$L \in \{1, 2, 3, 4, 5\}$$

with values as defined:

$L = 1$: Rare (Very Unlikely)
$L = 2$: Unlikely
$L = 3$: Possible (Moderately Likely)
$L = 4$: Likely
$L = 5$: Almost Certain (Very Likely)

- I represent the **Impact** score, where:

$$I \in \{1, 2, 3, 4, 5\}$$

with values as defined:

$I = 1$: Insignificant
$I = 2$: Minor
$I = 3$: Moderate
$I = 4$: Major
$I = 5$: Catastrophic

The **Risk Score** is calculated using the following formula:

$$\text{Risk Score} = L \times I$$

Risk Tolerance Zones:
- Critical Risk : Risk Score 16 - 25
- High Risk : Risk Score 10 - 15
- Moderate Risk : Risk Score 5 - 9
- Low Risk : Risk Score 1 - 4

Table 5.1: Risk Assessment

Risk Number	**Risk**	**Likeli-hood**	**Impact**	**Conditions**
1 Mitigation 1	When a foundation assigns a multiplier for a repository, the developer of that repository could donate 120$ by himself and gets 240$ in return.	Almost Certain	Catas-trophic	- the developer must be alone contributing to the repository - the developer only selects his own repository for donation - the repo has a multiplier assigned by a foundation
10 Mitigation 10	The developer could just ignore our terms and conditions even though he has a connection to the donation. Therefore the attacker has still the same opportunities. (same vulnerability as Risk 1)	Possible	Catas-trophic	-
11 Mitigation 11	This developer could just set up another repository, it doesn't matter whether this is a serious repository or not. As long as there are two repos selected, one multiplier will be applied.	Possible	Catas-trophic	- the developer must be alone contributing to both repositories - the developer only selects his two repository for donation - the repo has a multiplier assigned by a foundation
12 Mitigation 12	As a malicious developer I would still be able to drain a foundation. As described in the diagram 14 and 15 in mitigation example 2, with a payment of $120 I am able to get $120 back. I would not benefit from it, but I can still harm the foundation, by paying me an extremely high multiplier sponsoring.	Unlikely	Major	- the developer must be alone contributing to the unverified repository - the developer only selects his own repository for donation - the repo has a multiplier assigned by a foundation

Risk Number	Risk	Likelihood	Impact	Conditions
13 Mitigation 13	As a malicious developer I could do my best to gain a "verified stamp" on one of my repositories. Because no restrictions are applied to verified repositories when sponsored alone, I can now benefit from a higher payout than the original payment.	Unlikely	Major	- the developer must be alone contributing to the repository - the developer only selects his own repository for donation
14	The Attacker can evade our identification system.	Unlikely	Major	-
20 Mitigation 20, 21	Fraudulent transactions: Payment with a stolen credit card	Possible	Moderate	-
21	Fraudulent transactions: A legitimate cardholder disputes a transaction to claim a refund (aka charge-back fraud).	Possible	Major	-

Table 5.2: Likelihood and Impact Investigation

Risk Number	Likelihood	Impact	Risk Score
1	This vulnerability could allow a user to drain the foundation's resources. Even an ordinarily law-abiding person might be tempted to exploit it. Many would seize the opportunity to get rich if it were handed to them on a silver platter. $\rightarrow$ Almost Certain	Such exploitation could lead to massive financial loss of the foundation, and a decrease in user trust. Once exposed, the platform's reputation and future viability would be severely compromised. $\rightarrow$ Catastrophic	25
10	The attacker must have the audacity to violate the Terms and Conditions, thereby committing a criminal offense. $\rightarrow$ Possible	If exploited, same impact as Risk 1. $\rightarrow$ Catastrophic	15

Risk Number	Likelihood	Impact	Risk Score
11	The attacker would need a greater level of criminal intent, actively creating a new account. $\rightarrow$ Likely	If exploited, same impact as Risk 1. $\rightarrow$ Catastrophic	15
12	The attacker must have malicious intent, as their actions provide no personal gain and serve only to harm the foundation. $\rightarrow$ Unlikely	The maximum amount exploitable per day is capped at the foundation's daily limit. To some extent, the foundation's own policies influence the severity of this risk. $\rightarrow$ Major	8
13	If a developer chooses to drain a foundation, they may also attempt to acquire a verified repository. This likelihood directly refers to the false positive rate of our Verified sign. $\rightarrow$ Unlikely	If exploited, the attacker would gain significantly; however, due to the foundation's daily limit, the impact is not as severe as in Risk 1. $\rightarrow$ Major	8
14	The Likelihood some comes that far and is also able to evade the identification system is rare. $\rightarrow$ Unlikely	If exploited, same Risk as 13. $\rightarrow$ Major	8

5.1.2 Risk Mitigation

After identifying risks, we are able to define the counter measures. Some of them will be explained in detail in the next sections.

Table 5.3: Risk Mitigation Table

Risk Number	Mitigation Strategy	New Assessment
1	We create an algorithm to asses repositories on a scale from 1 - 10 based on various Git metrics etc. This so called Health Value helps an admin to decide, whether a repository should be marked as Verified Repository or not. If a repository is verified by the FFS Application, all assigned multipliers will be applied immediately. Therefore we only have to investigate what to do with Unverified Repository.	This is actually a big part of our solution and not further analyzed in the Risk Analysis. All the details will be covered in a separate section.

Risk Number	Mitigation Strategy	New Assessment
1	We define terms and conditions for our platform. Each User of our platform agrees to the Terms and Conditions by using our platform. see Terms and Conditions	Risk 10
10	When a sponsor adds only one unverified repository to his Sponsoring Bundle, none of the assigned multipliers will be applied to the repository. When a sponsor adds two unverified repositories to his Sponsoring Bundle only one multiplier will be applied, and so on. The main goal of this mitigation is, that unverified repository can't get more then the initial donation. see mitigation example 1	Risk 11
11	We make significant adjustments to the mitigation strategy for unverified repos. (1) As long as no verified repo is in the sponsoring bundle, no multiplier will be applied. (2) The amount of money that flows to verified repos is considered a **pool** with "verified money", which can be distributed among all the repos with one or more multipliers applied. The calculation of the distribution works as follow: 1. The amount of multipliers in the whole bundle are considered as **parts** 2. The value of one part is equal to the **pool** divided by the number of **parts**. 3. Each repository receives a part for each assigned multiplier. In this case, verified and unverified repos are treated the same. see mitigation example 2	Risk 12
12	We enhance our system with a maximum multiplier of 1.9. So we have to introduce a Payout Limit for multiplier sponsoring, which is exactly the amount one repo gets from the sponsor times 0.9. This 0.9 multiplier is crucial for the case with just one verified and one unverified repo in the bundle. see mitigation example 3	Risk 13
12	Here, we additionally offer the foundations a daily Multiplier Sponsoring Limit. Every foundation can set it's own boundary.	Risk 13
13	We create an identification system to determine the likelihood a fraudulent transaction got approved although the repo is verified.	Risk 14
20, 21	Use Stripe as payment service (already existing security decision). Stripe has very strong built in fraud prevention like 3D secure (3DS2). To make Stripe even more secure against fraud Stripe's Radar could be implemented.	-

5.1.3 Examples

In this subsection we try make it easier to follow our thoughts about the fraud techniques with (visual) examples.

Initial Situation of the Examples
- The Attacker makes a donation of **$120**

Mitigation Example 1

	repo A	**repo B**	**repo C**	**repo D**
amount of assigned multipliers by foundations or FFS sponsors to the repository	1	2	3	4

	sponsor selects only repo A	sponsor selects repo A and B	sponsor selects repo A, B and C	sponsor selects repo A, B, C, D
cash for repo A from sponsor	$120	$60	$40	$30
cash for repo A from sponsor + multiplier sponsoring from foundation(s)	$120	$120	$80	$60
cash for repo B from sponsor		$60	$40	$30
cash for repo B from sponsor + multiplier sponsoring from foundation(s)		$120	$120	$90
cash for repo C from sponsor			$40	$30
cash for repo C from sponsor + multiplier sponsoring from foundation(s)			$120	$120
cash for repo D from sponsor				$30
cash for repo D from sponsor + multiplier sponsoring from foundation(s)				$120

Mitigation Example 2

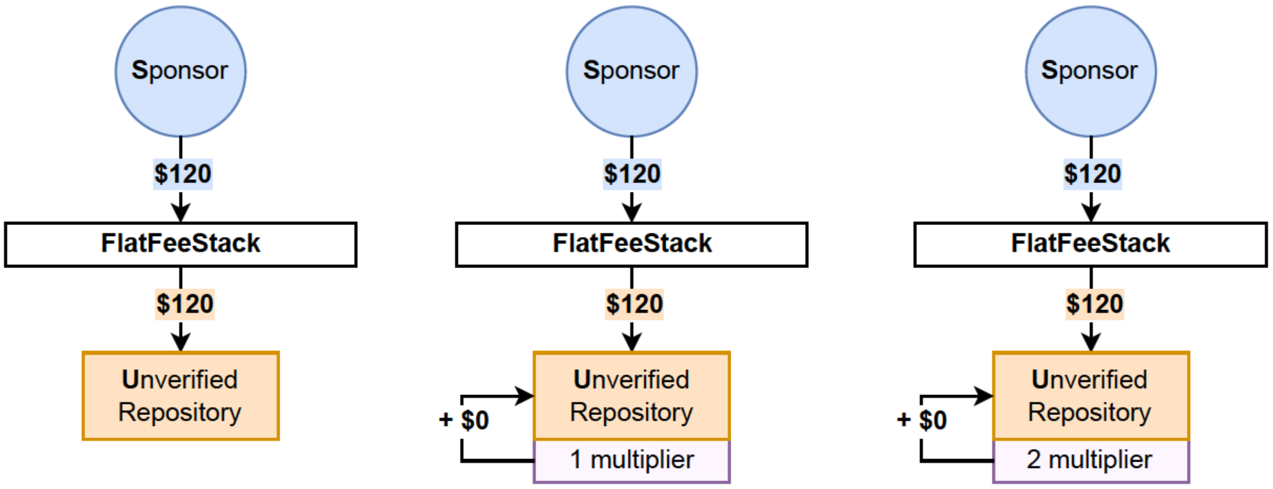


Figure 5.1: Sponsoring of one single unverified repo - diagram 1 - 3

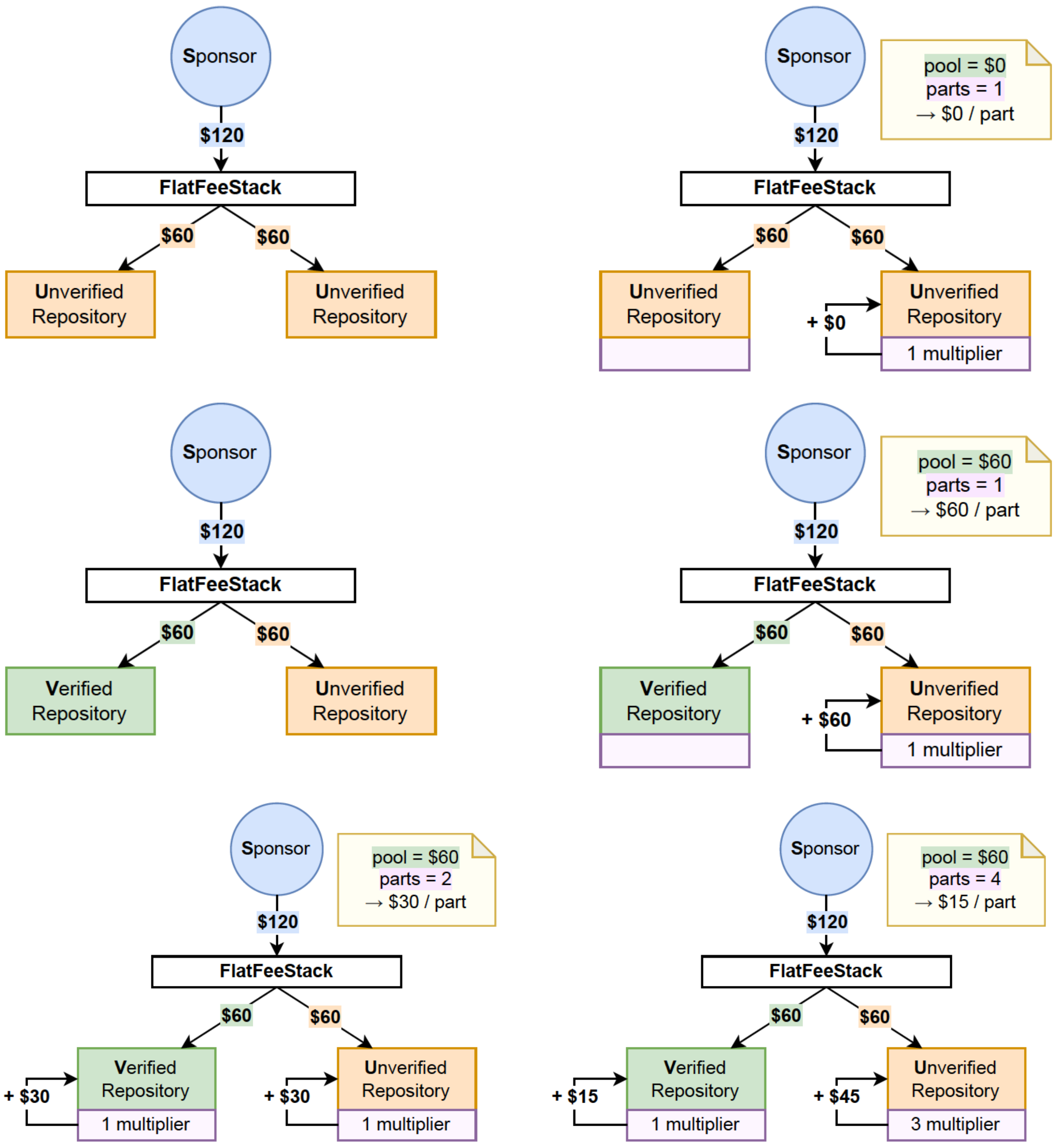


Figure 5.2: Sponsoring of two repos in a heterogeneous bundle - diagram 4 - 9

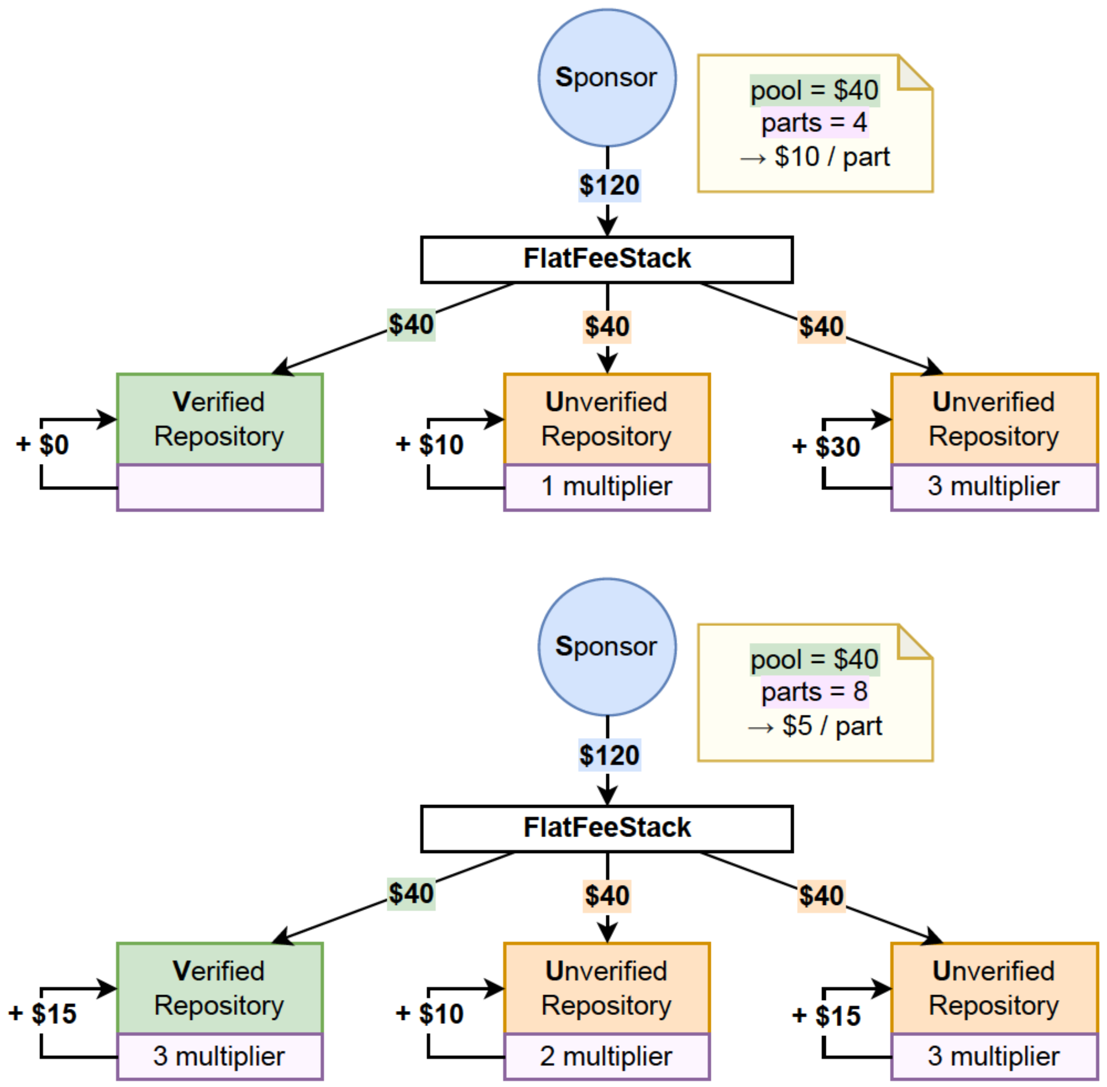


Figure 5.3: Sponsoring of three repos in a heterogeneous bundle - diagram 10 - 11

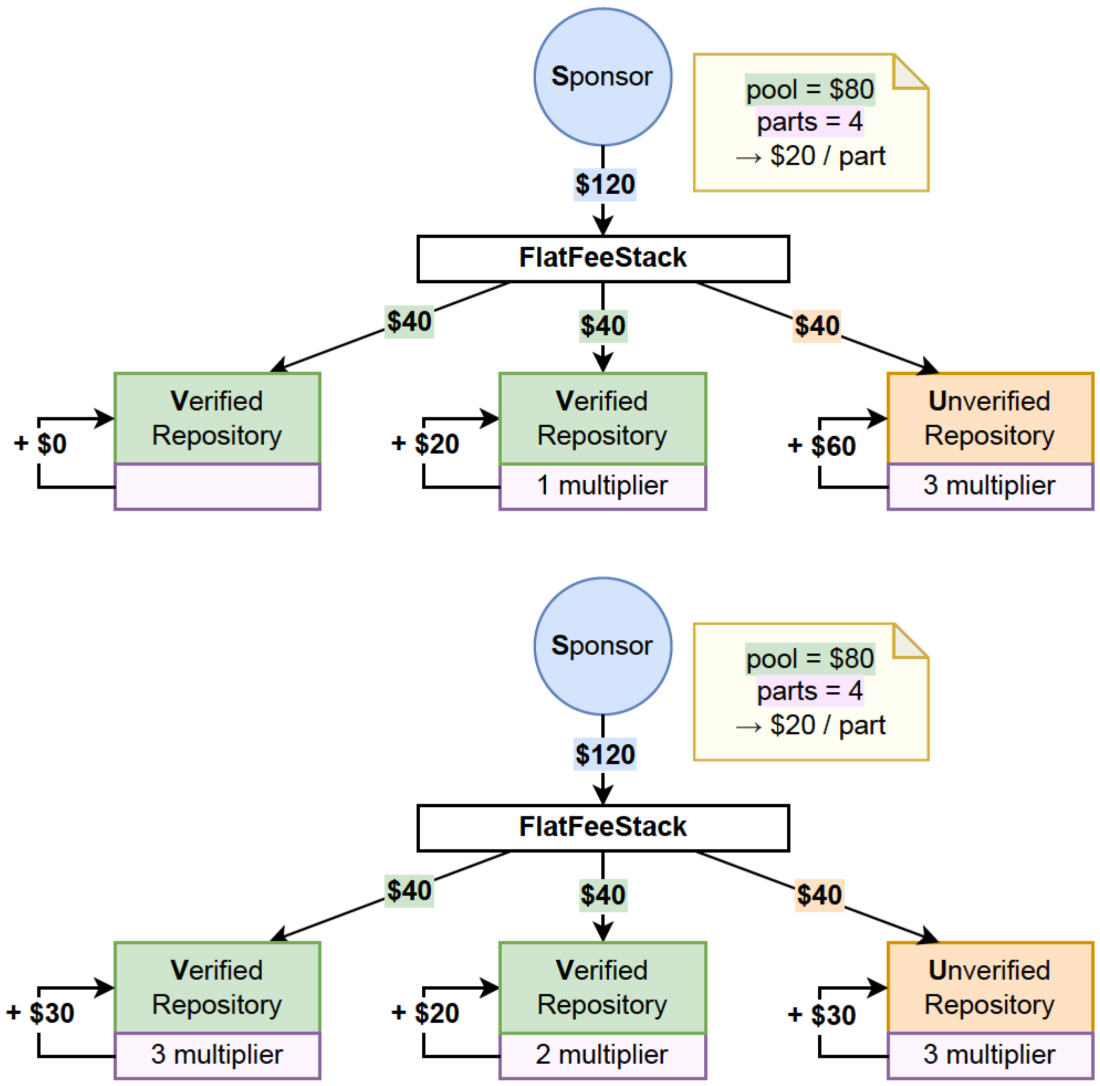
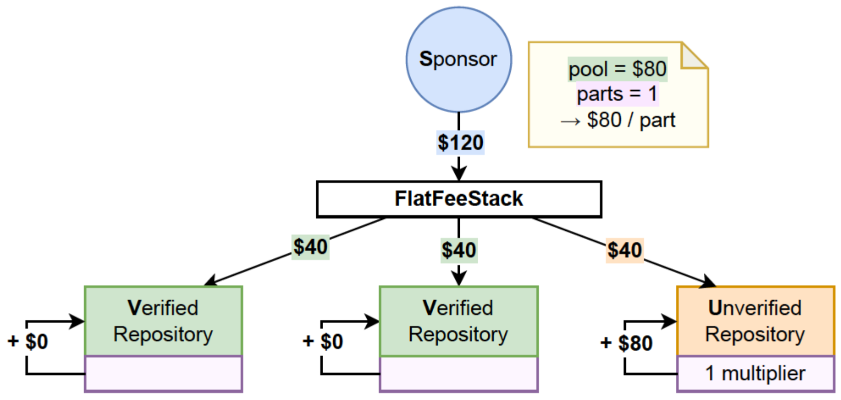


Figure 5.4: Sponsoring of three repos in a heterogeneous bundle - diagram 12 - 13

The last two diagrams, Diagrams 14 and 15, are intended solely to illustrate the worst-case scenario in more detail. The worst case occurs when only one unverified repository is selected, and the rest are verified repositories without a multiplier. This scenario is also depicted in Diagram 7, where two repositories are selected, one verified and one unverified.



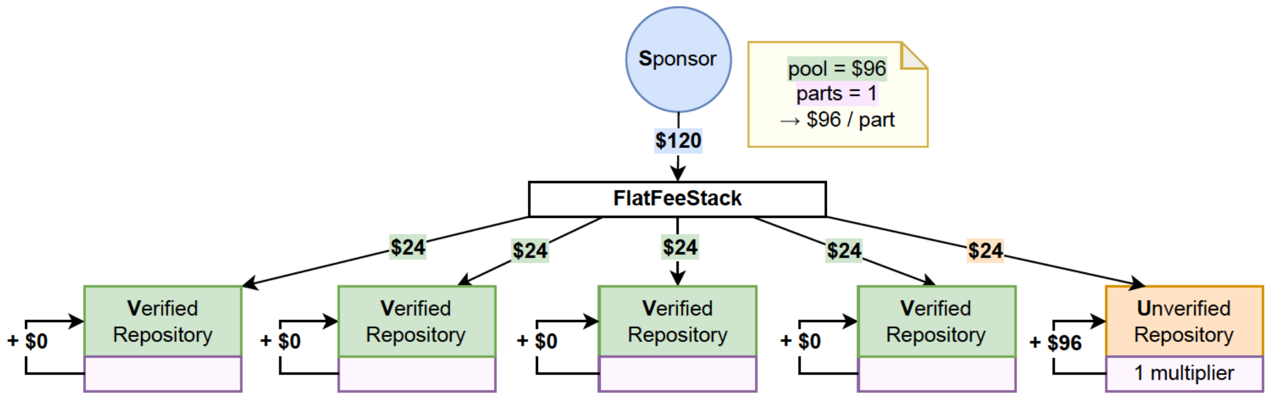


Figure 5.5: Sponsoring of five repos in a heterogeneous bundle - diagram 14 - 15

Mitigation Example 3

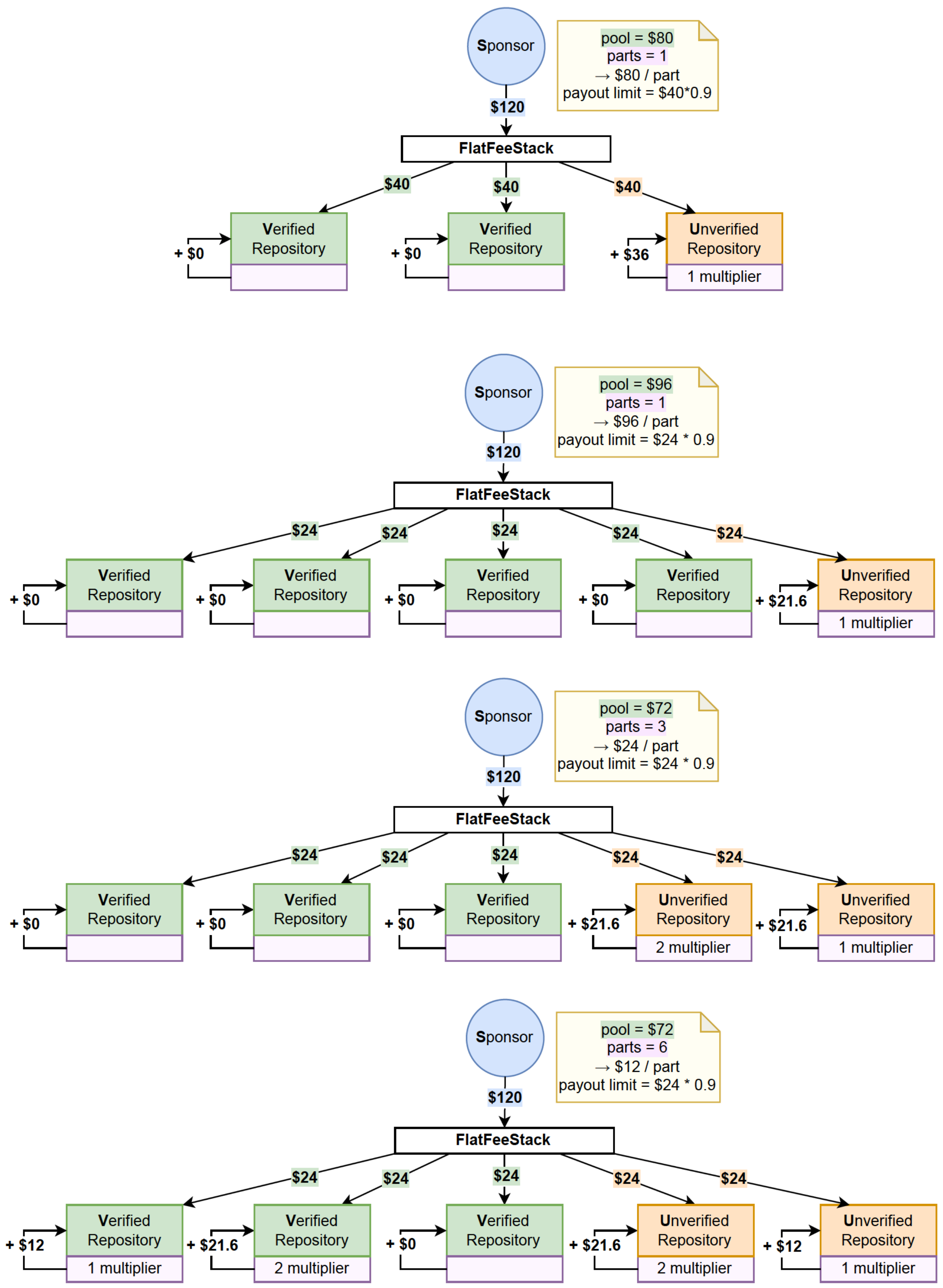


Figure 5.6: Sponsoring of five repos in a heterogeneous bundle - diagram 16 - 19

5.2 Fraud Prevention Techniques

Following the identification of the potential fraud techniques we delve into various fraud prevention techniques. While some of them will be explained in detail in the next sections, the rest of them are not necessary. Suffice to say, even with all the best techniques fully preventing fraud is a tall task to accomplish. Thus our main goal is to create a solution, to mitigate the risk of fraud.

5.2.1 Terms and Conditions

This approach may not directly mitigate the risk of fraud. However, it serves as a deterrent by signaling to potential wrongdoers that we take these matters seriously. By clearly outlining the consequences of fraudulent behavior in our terms and conditions (AGB), we aim to discourage any developers who might be considering such actions.

---

1. Introduction
Welcome to our open-source donation platform. By using this platform, you agree to the following terms and conditions, which govern your access to and use of the platform, as well as your participation in any donation activities related to open-source projects. Our platform enables users to donate to open-source Git repositories, with an option to apply a multiplier to specific repositories. This means that when a user donates to a project, the foundation/user will donate based on a predefined rule (see source code).

By using our platform, you agree to these Terms and acknowledge the associated guidelines and restrictions.

2. Platform Usage

- The platform is intended for the sole purpose of supporting open-source projects by facilitating donations.
- By creating an account on the platform, users agree to act in good faith when donating to open-source repositories.

3. Multiplier Feature

- A foundation or any user may choose to apply a multiplier to specific Git repositories. When a multiplier is active, the foundation/user will donate based on a predefined rule (see source code) and the donation made to that project by a user.
- Users are prohibited from any fraudulent activities intended to exploit the multiplier system, including but not limited to:
  - Self-donating: Developers or project maintainers are not permitted to donate to their own repositories with the intent of receiving funds from the foundation/user.

    - Collusion: Users must not conspire with others to falsely inflate donations with the goal of benefiting from the multiplier.

4. Fraudulent Activities

- Prohibited behavior: Users agree not to engage in any behavior that could be considered fraudulent, deceptive, or intended to manipulate the donation process for personal gain. This includes:
    - Creating multiple accounts to falsely increase the amount donated to a specific repository.
    - Donating to repositories in which they have a direct financial interest, with the intent of abusing the multiplier feature.

5. Sanctions for Fraudulent Behavior

- If fraudulent activity is detected, the platform reserves the right to:
    - Suspend or terminate the user's account.
    - Reclaim any matched funds from the foundation/user that were obtained through fraudulent means.
    - Notify the foundation/user of the fraudulent activity and any involved repositories or users.

6. User Responsibilities

- Honesty: Users agree to donate in good faith and without any intent to deceive or manipulate the platform, the foundation, or other users.
- Transparency: Users must ensure that their actions comply with these Terms and should report any suspicious activity or potential misuse of the platform.

7. Legal Actions

- Users acknowledge that any violation of these Terms may result in legal action. Fraudulent activities will not only lead to account termination but could also result in civil or criminal charges.

8. Modification of Terms

- The platform reserves the right to modify these Terms at any time. Users will be notified of any significant changes, and continued use of the platform constitutes acceptance of the updated Terms.

9. Governing Law

- These Terms are governed by the laws of Switzerland, without regard to its conflict of laws principles. Any disputes arising out of or in connection with these Terms will be resolved in the courts of Switzerland.

5.2.2 Fraud prevention for Unverified Repos

We have already created some sketches to understand our approach to handling multiplier sponsorships for unverified repositories. Now, we need to develop a plan to guide us through the implementation of the multiplier feature. This requires further investigation into how we intend to accomplish these functionalities.

Applying Multiplier Process

To illustrate the entire process, we use a practical example:

Legend / Sponsorships

S1 = Sponsor 1 → donates \$120

S2 = Sponsor 2 → donates \$120

T1 = Verified Repository 1

T2 = Verified Repository 2

U1 = Unverified Repository 1

U2 = Unverified Repository 2

U3 = Unverified Repository 3

F1 = Foundation 1 → has multiplier sponsoring limit of \$50

F2 = Foundation 2 → has multiplier sponsoring limit of \$50

MMS1 = Multiplier Money from Sponsoring 1

MMS2 = Multiplier Money from Sponsoring 2

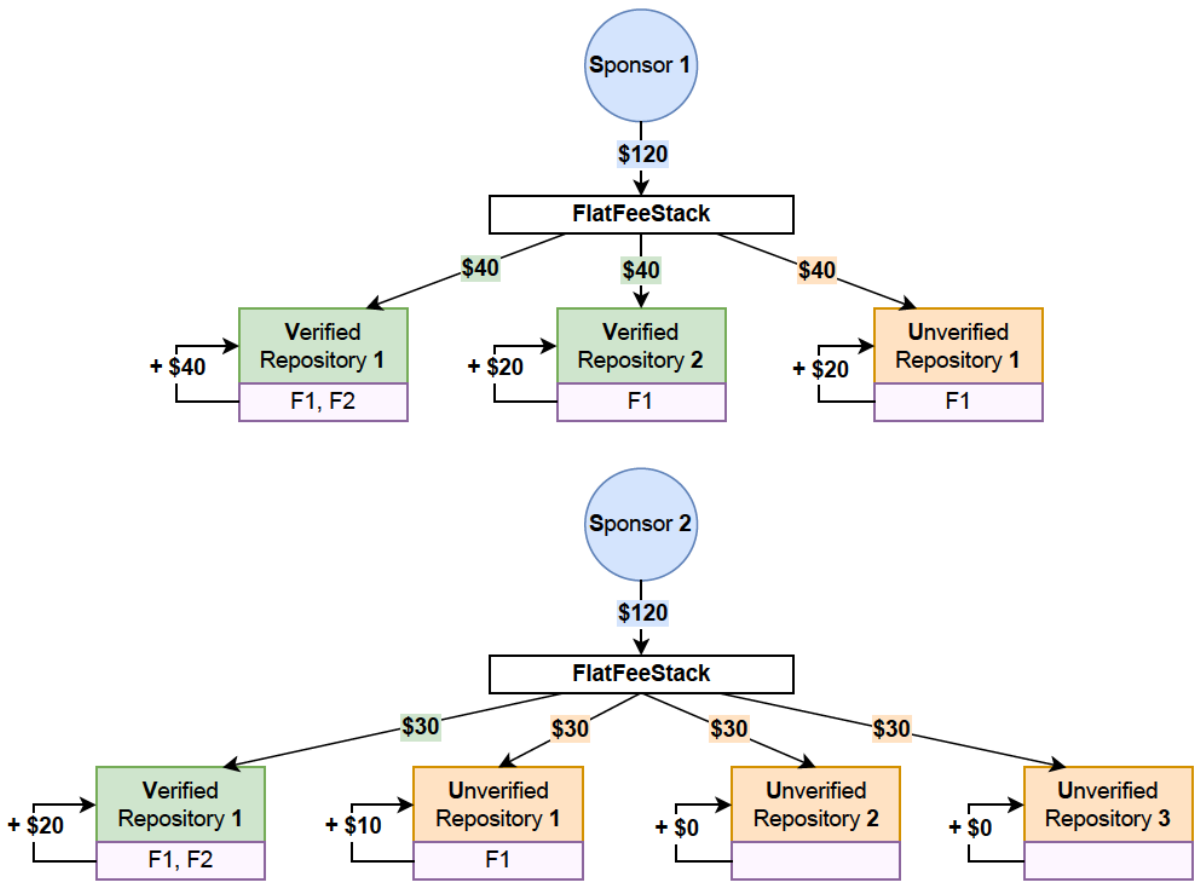


Figure 5.7: sponsoring example for process analysis

1. Verified Money Pool Calculation with Payout Limit

The logic of **verified money pool** and **parts** is already very well documented in Risk Mitigation at Risk Number 10. With pseudocode it would look like this:

The 0.9 rule is taken into account in the function *getVerifiedMoneyPool*:

Algorithm 1: getVerifiedMoneyPool

```
Input: starredReposOfUser *RepoList
Input: sponsorAmount *float
Output: float (Verified Money Pool)
/* Initialize the amount of verified repositories */
amountOfVerifiedRepos ← 0;
/* Iterate through each repository in the starred list */
foreach repo ∈ starredReposOfUser do
    if repo.verified then
        amountOfVerifiedRepos ← amountOfVerifiedRepos + 1;
/* Calculate money per repository */
amountPerRepo ← sponsorAmount / starredReposOfUser.length;
/* Calculate the total verified money */
verifiedMoney ← amountPerRepo * amountOfVerifiedRepos;
/* Apply the 0.9 rule to the verified money */
verifiedMoneyPointNineRule ← verifiedMoney * 0.9;
return verifiedMoneyPointNineRule;
```

Algorithm 2: getAmountOfPartsInPool

```
Input: starredReposOfUser *RepoList
Output: int (Amount of Parts in Pool)
/* Initialize the count of parts */
parts ← 0;
/* Iterate through each repository in the starred list */
foreach repo ∈ starredReposOfUser do
    if repo.hasMultiplier then
        parts ← parts + repo.multiplier.length;
return parts;
```

Algorithm 3: getPartsValue

```
Output: float (Value of Each Part)
/* Retrieve the verified money pool value                          */
pool ← getVerifiedMoneyPool();
/* Retrieve the total number of parts in the pool                  */
parts ← getAmountOfPartsInPool();
/* Calculate the value of a single part                            */
partValue ← pool / parts;
return partValue;
```

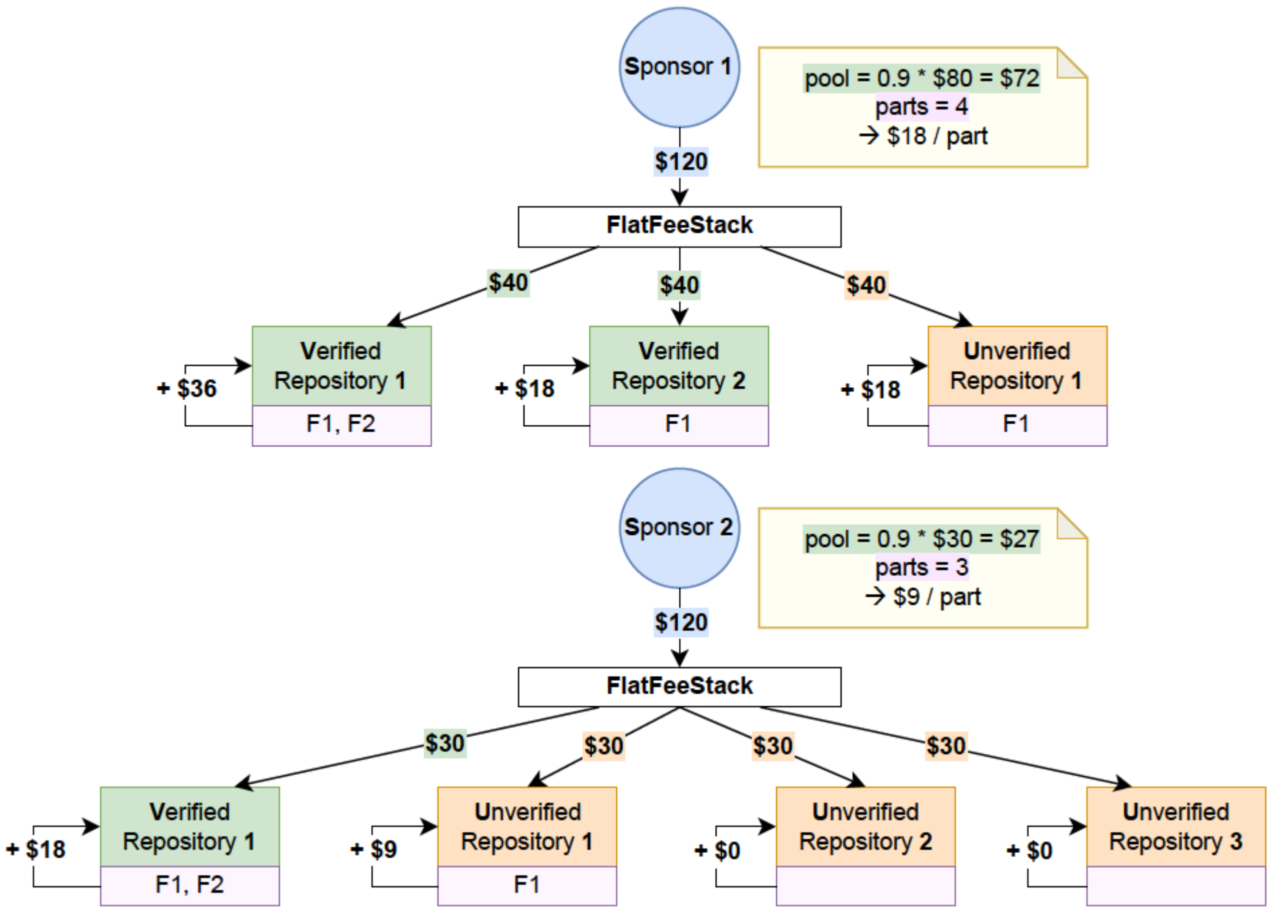


Figure 5.8: Verified Money Pool including the 0.9 rule on example

2. Multiplier Sponsoring Calculation per Foundation without Multiplier Sponsoring Limit

Table 5.4: calculation F1 (without multiplier sponsoring limit)

	T1	T2	U1	U2	U3	
MMS1	$18	$18	$18			$54

	T1	T2	U1	U2	U3	
MMS2	\$9		\$9			\$18
repo weight	\$27 $\downarrow$ = $\frac{27}{72}$ = $\frac{3}{8}$	\$18 $\downarrow$ = $\frac{18}{72}$ = $\frac{2}{8}$	\$27 $\downarrow$ = $\frac{27}{72}$ = $\frac{3}{8}$			\$72

Table 5.5: calculation F2 (without multiplier sponsoring limit)

	T1	T2	U1	U2	U3	
MMS1	\$18					\$18
MMS2	\$9					\$9
repo weight	\$27 $\downarrow$ = $\frac{27}{27}$ = $\frac{1}{1}$					\$27

3. Multiplier Sponsoring Calculation per Foundation with Multiplier Sponsoring Limit

Algorithm 4: setPayFactor

```
Input: dailyMultiplierSponsoringLimit *float,
       dailyMultiplierSponsoringSum *float
Output: float (Pay Factor)

/* Check if the daily multiplier sponsoring limit is greater than or
   equal to the daily multiplier sponsoring sum                      */
if dailyMultiplierSponsoringLimit $\geq$ dailyMultiplierSponsoringSum then
    setPayFactor(1);
    /* Set pay factor to 1 if within the limit                       */
else
    setPayFactor(dailyMultiplierSponsoringLimit / dailyMultiplierSponsoringSum);
    /* Scale pay factor proportionally if limit is exceeded          */
```

Algorithm 5: dailyMultiplierSponsoringCalculation

```
Input: dailyRepoPaymentSum *float, payFactor *float
Input: reposWithMultiplier *RepoList
Output: RepoList (Updated with Multiplier Sponsoring Values)
/* Retrieve repositories associated with the foundation           */
reposWithMultiplier ← getReposOfFoundation();
/* Update multiplier sponsoring for each repository               */
foreach repo ∈ reposWithMultiplier do
    repo.multiplierSponsoring ← dailyRepoPaymentSum * payFactor;
return reposWithMultiplier;
```

Foundation 1 - Calculation with Multiplier Sponsoring Limit

In the case of F1 the multiplier sponsoring limit is lower then the daily multiplier sponsoring sum, so the else clause will be applied as follow:

Algorithm 6: setPayFactor Foundation 1

```
Input: dailyMultiplierSponsoringLimit = 50, dailyMultiplierSponsoringSum =
        72
Output: float (Pay Factor)
/* Check if the daily multiplier sponsoring limit (50) is greater
   than or equal to the daily sum (72)                              */
if $50 \geq 72$ then
    setPayFactor(1);
    /* Set pay factor to 1 if within the limit                      */
else
    setPayFactor(50 / 72); /* Scale pay factor proportionally:  50/72   */
```

In the second step the received multiplier sponsoring each repo gets from Foundation 1 is calculated:

Algorithm 7: dailyMultiplierSponsoringCalculation Foundation 1

```
Input: dailyRepoPaymentSum = {27, 18, 27}, payFactor = 50/72
Input: reposWithMultiplier = [T1, T2, U1]
Output: RepoList (Updated with Multiplier Sponsoring Values)
/* Define the list of repositories with their respective payment
   sums and pay factors                                             */
reposWithMultiplier ← [T1, T2, U1];
/* Update multiplier sponsoring for each repository                 */
T1.multiplierSponsoring ← (50 / 72) * 27 = 18.75;
T2.multiplierSponsoring ← (50 / 72) * 18 = 12.5;
U1.multiplierSponsoring ← (50 / 52) * 27 = 18.75;
return reposWithMultiplier;
/* Return updated repository list                                   */
```

Foundation 2 - Calculation with Multiplier Sponsoring Limit

In the case of Foundation 2 the multiplier sponsoring limit is higher then the daily multiplier sponsoring sum, so the if clause will be applied as follow:

```
Algorithm 8: setPayFactor Foundation 2
Input: dailyMultiplierSponsoringLimit = 50, dailyMultiplierSponsoringSum =
       27
Output: float (Pay Factor)

/* Check if the daily multiplier sponsoring limit (50) is greater
   than or equal to the daily sum (27)                               */
if $50 \geq 27$ then
    setPayFactor(1); /* Set pay factor to 1 since limit is not exceeded
        */
else
    setPayFactor(50 / 27);
    /* Scale pay factor proportionally:  50/27                       */
```

In the second step the received multiplier sponsoring each repo gets from Foundation 2 is calculated:

```
Algorithm 9: dailyMultiplierSponsoringCalculation Foundation 2
Input: dailyRepoPaymentSum = {27}, payFactor = 1
Input: reposWithMultiplier = [T1]
Output: RepoList (Updated with Multiplier Sponsoring Values)

/* Define the list of repositories with their respective payment
   sums and pay factors                                              */
reposWithMultiplier ← [T1];

/* Update multiplier sponsoring for each repository                  */
T1.multiplierSponsoring ← 1 * 27 = 27;

return reposWithMultiplier;
/* Return updated repository list                                    */
```

5.2.3 Fraud Prevention Options for Unverified Repos

Technique	Description	Fraud prevention Effectiveness	Scope
Manual	As the name implies this is a manual selection of verified repositories.	High	FR
Health Value	The Health Value is calculated based on defined criterias. These can be found here. Initially the health Value has low credibility.	Low - High	FR
Sugges-tions	Foundations might have the option to suggest repositories. If repositories are suggested often enough it will be highlighted to other foundations or suggested for manual selection of verified repositories.	Medium	NFR
Voting Based	Voting Based is similar to suggestions with the difference being, that foundations will regularly have an option to vote for repositories appear.	Medium	NFR

Table 5.6: Fraud Prevention Techniques

5.2.4 Health Value: Definition

In its inception the Health Value is comprised of six metrics, two external and four internal. For simplicity sake the Health Value is depicted numerically on a scale form 0 to 10. Its metrics will be stored as raw values and based on dynamically calculated. In the calculation process the metric will be mapped against a Threshold with an upper and lower bound and an associated weight to each metric. The result delivers 6 values which can sum up to a maximum of 10. Each individual component of the calculation result is called PartialHealthValue

To make the distribution of the points transparent and clear, there is a limit to how much one Partial Health Value contributes to the resulting HealthValue. This is expressed by a Weight Value.

Active User Significance Active Users are crucial for maintaining the integrity of repository metrics and serve as a fraud prevention mechanism, ensuring that users cannot manipulate metrics (e.g., creating fake accounts to artificially inflate stars or sponsorships). By focusing on donations to verified repositories, this approach enhances the reliability of repository evaluation metrics.

ContributorCount: Active contributors a repository has in the last three months

This metric is purely external. The sources for this metric are git repos. Depending on the size of a repositories the number of contributors can be highly misleading. Thus we've taken the number of active contributors from the last three months. Based on the three month average a metric can be derived. With this data analyzed thresholds can be defined.

CommitCount: Amount of commits a repository has in the last three months

This metric is also external just like the previous one. Analyzing the cumulative amount of commits in the last three months a robust metric can be derived. The previous metric and this metric together will be the only external values contributing the the Health Value.

SponsorCount: Sponsoring Count

Projects that have received a lot of donations should be regarded as more legitimate than completely unknown projects, as it is unlikely that a person or small group will donate so much to themselves to achieve more points in this metric.

MultiplierCount: Multiplier Sponsoring Count

The amount of active Foundations, with the same conditions like the Sponsoring Count before, set a Multiplier to a specific repo. Only active foundations will count towards this metric. Activity is measured by sponsorships.

StarCount: Star Count

The amount of stars a repo on FlatFeeStack has. Regardless if the user is an active contributor. This metric is counted if the last contribution is at most three months in the past.

ActiveFFSUserCount: Active FFS User Count

Developers actively participating in FFS and contributing to other repos, will see their repos gain in this metric. This is an incredible valuable metric increasing the trustworthiness in this developer and subsequently their git repo.

PartialHealthValue: Partial Health Value

Another important concept used for the Health Value is the PartialHealthValue. As the HealthValue comprises of different metrics calculated together, each individual metric delivers a PartialHealth-Value upon application of the PartialHealthValue Algorithm.

Metric Weight

Metric Weight helps fine tune every PartialHealthValue. Each metric has a weight associated with it. The weight explains how significant the partial value to the final HealthValue is. The weights together add up to 10 in total, as our chosen scale is from 0 to 10.

Threshold

The Threshold is the final component required for calculating the Health Value. It consists of a lower and upper bound, between which the raw metric value is linearly mapped. Both the lower and upper bounds are inclusive, meaning values equal to these bounds are included in the calculation.

The algorithm calculates all provided parts into one single value. All six components described as input values, will result in the PartialHealthValue.

5.2.5 Fee Multiplier

The Fee Multiplier offers three potential implementation approaches:

1. Automatic daily credit card withdrawals
2. Automatic weekly/monthly credit card withdrawals
3. Manual balance top-up with daily deductions

While the first two options appear similar, their implications differ significantly. Weekly or monthly withdrawals would reduce transaction fees compared to daily withdrawals. However, beyond cost considerations, there are other compelling reasons why we chose not to pursue automatic withdrawal options.

The primary reason for rejecting automatic withdrawals centers on security considerations. Such a system would require FlatFeeStack to permanently store sensitive credit card information in its database. Although this is standard practice in the industry, the potential consequences of a data breach or system compromise present an unacceptable risk level for FlatFeeStack's operational model. A secondary consideration stems from our existing daily contribution system. Since our platform already operates on a daily distribution cycle, the manual top-up model naturally aligns with our core functionality, making it the most advantageous approach.

By implementing the manual top-up system, foundations maintain a balance pool from which FlatFeeStack makes daily deductions. This approach significantly reduces security risks by eliminating the need to store payment credentials while maintaining efficient daily distributions. The specific deduction amounts for each repository follow the calculation method detailed earlier in this section.

Chapter 6

Implementation

This chapter presents the key implementations in the backend, focusing on two core functionalities: the calculation of the Health Value and the fee multiplier. To develop a reliable Health Value, an extensive analysis of a large number of diverse repositories was conducted. Consequently, this chapter also covers the processes of data collection, mining, and analysis that underpin these functionalities.

6.1 Calculation of Health Value

Algorithm 10: CalculatePartialHealthValues

```
Input: weights *Thresholds
Input: threshold *Thresholds
Input: metrics *RepoHealthMetrics
/* Calculate partial health values for each metric                    */
Output: map[string]float64

/* Initialize result map                                              */
result ← new(PartialHealthValues);

/* Calculate health values for each metric                            */
result.val1 ←
    calcValue(metrics.val1, threshold.val1.Lower, threshold.val1.Upper, weights.val1);
result.val2 ←
    calcValue(metrics.val2, threshold.val2.Lower, threshold.val2.Upper, weights.val2);
result.val3 ←
    calcValue(metrics.val3, threshold.val3.Lower, threshold.val3.Upper, weights.val3);
result.val4 ←
    calcValue(metrics.val4, threshold.val4.Lower, threshold.val4.Upper, weights.val4);
result.val5 ←
    calcValue(metrics.val5, threshold.val5.Lower, threshold.val5.Upper, weights.val5);
result.val6 ←
    calcValue(metrics.val6, threshold.val6.Lower, threshold.val6.Upper, weights.val6);

return result;
```

The two external metrics are derived from analyzing public repository statistics. While these metrics contribute to the Health Value, they have a lower impact compared to other factors, as they are not

Algorithm 11: calcValue

```
Input: int metric, lower, upper, weight
Output: float
if $metric < lower$ then
    return 0.0;
else if $metric > upper$ then
    return float64(weight);
else
    diff ← upper - lower + 1;
    normalized ← metric - lower + 1;
    return round(100*(weight/diff)*normalized) / 100;
```

considered the most significant indicators of repository health. These external metrics account for 40% of the total Health Value calculation.

6.2 Health Value: Data Collection, Mining and Analysis

This section describes how the data is gathered and mined. The first part discusses the data collection and mining form external sources. Afterwards the analysis and implementation of the external and internal data is presented. This section is highly important to building the complete Health Value.

6.2.1 Data Collection: External Sources

The two metrics discussed are "Active Contributors" and "Amount of Commits". The process of gathering the data is fairly simple. For simplicity github.com and gitlab.com were used as data sources. Both sites provide a simple API endpoint to gather the necessary data. The only limiting factor are the rate limits imposed by github and gitlab.

Repos	**github**	**gitlab**
Endpoint	`https://api.github.com/repos`	`https://gitlab.com/api/v4`
Search Path	/search/repos	/projects
User Path	/repos/{owner}/{repo-name}/contributors	/projects/{owner}%2F{repo-name}/repository/contributors
Commit Path	/owner/repo-name/commits	/projects/{owner}%2F{repo-name}/repository/commits

Table 6.1: Git Repo Endpoints

The next point showcases an example query of a github api endpoint. The rest of the endpoints are queried in a similar fashion and will therefore not be described. The limitation here is a rate limit. FlatFeeStack already has a functionality to analyze repos. But for this specific task we opted to write our own scripts to query the endpoints. All necessary scripts can be found in the appendix or in the GitLab repo for the documentation.

curl example of api query

```
$> curl -s https://api.github.com/repos/flatfeestack/\
> flatfeestack/commits?per_page=1 | jq .[0].commit.committer
{
  "name": "GitHub",
  "email": "noreply@github.com",
  "date": "2024-10-19T00:30:31Z"
}
```

6.2.2 Analysis of External Data

The most reliable analysis comes from organizing the data into bins based on user activity. By examining the average daily commits per user within each bin, we obtain a comprehensive view of repository activity. The analysis considers three key dimensions:

- The number of active contributors, grouped into distinct ranges
- Repository size classification
- Development activity measured through commit frequency

The binning analysis of our datasets has provided the most significant insights into repository distribution patterns. The data demonstrates a substantial concentration in smaller repositories, with a marked margin compared to other size ranges. Notably, while these smaller repositories dominate the distribution, they exhibit the lowest average daily commit rate per user - an observation we consider plausible.

FlatFeeStack's fundamental mission is to support open-source development across all scales, from small individual projects to large collaborative efforts. Our analysis strongly indicates that smaller projects require additional incentivization mechanisms. Consequently, we will adjust the CommitCount and ContributorCount thresholds to better accommodate and support these smaller-scale repositories.

The Kernel Density Estimation (KDE) analysis further validates these observations, revealing a bimodal distribution with a primary concentration around smaller repositories and a distinct, smaller cluster in the higher range. This distribution pattern reinforces our data-driven approach to threshold optimization.

Cluster Binning	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5
User Range	1 - 37	38-138	139 - 341	342 - 689	690 - 2364
Class	1	2	3	4	5
Repo Count	4266	334	98	6	14
Daily $\frac{count\ commit}{countcontrib}$	1.1	8.5	29.7	58.5	186.5

Table 6.2: bin user repo distribution

Number of Re-pos	**Mean**	**Median**	**Std Dev**	**Min**	**Max**
4720	22.18	3.00	123.02	1	2364
Five Number Summary Users	**Min**	**Q1**	**Median**	**Q3**	**Max**
4105	1	1	3	12	23
Five Number Summary Repos	**Min**	**Q1**	**Median**	**Q3**	**Max**
4105	1	8	31	135	320

Table 6.3: Summary of Statistics Overview

Repos: 4720	**Size**	**Mean**	**Median**	**Std Dev**
Cluster 1	4275	6.26	2	8.12
Cluster 2	13	2196.77	2181	59.5
Cluster 3	327	72.07	65	26.02
Cluster 4	104	232.87	216	71.17
Cluster 5	1	1562	1562	0

Table 6.4: Summary of Statistics Active Users

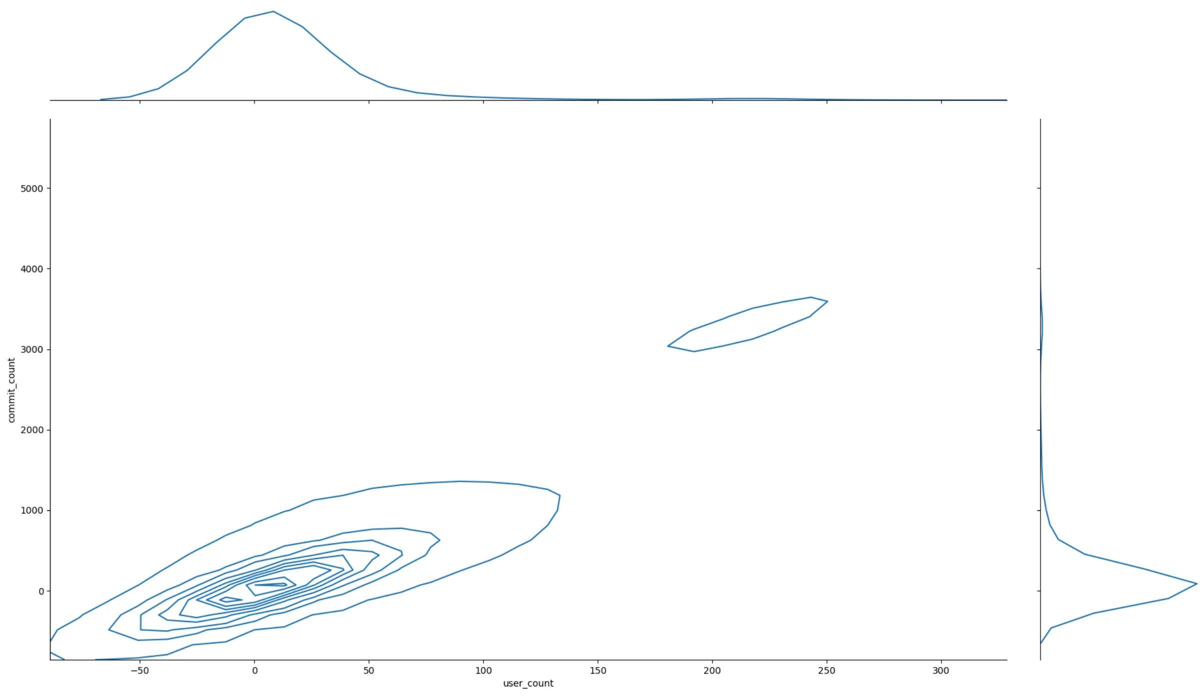


Figure 6.1: Health Value: Kernel Density Estimation

6.2.3 Analysis of Internal Data

The Health Value calculation incorporates four distinct internal metrics. Since there is no historical data available for internal repository evaluation and activity patterns, we have established initial threshold values based on deliberation and expert judgment.

While the Health Value plays a modest role in the platform's current operations, the four internal metrics will become increasingly valuable as FlatFeeStack grows. Their true significance will emerge once the platform establishes a robust user base and achieves higher activity levels.

6.2.4 Implementation of Partial Health Values

Each Metric results in a PartialHealthValue. All partial healt values together compose the Health Value.

External Partial Health Values Implementation

ContributorCount Implementation
There were very few changes required to implement analysis of this metric. As the Analyzer already gathers all related data to repos, the only task left to do was to count the amount of unique contributors in the last three months.

CommitCount Implementation
Similar to active contributors the Analyzer already gathered all necessary data. We've adjusted the

return payload to include a cumulative count for each repo analyzed.

Internal Partial Health Values Implementation

SponsorCount Implementation

This metric is fairly simple, as the sponsorships already paid on a particular repo are queried using a database select statement. This number can only increase as it does not look at a time period. The thresholds cannot simply be found out by means of tests, so they can be adjusted afterwards by the admin depending on the situation.

StarCount Implementation

The **Star Count** metric represents the total number of unique Active User, to be recognized as activeSponsors in the following algorithms, who have starred a repository. This indicates the visibility and appreciation of the repository in the developer community.

Algorithm 12: GetStarCount

```
Input: repoId: Repository ID, activeSponsors: Active Sponsor IDs
Output: int: Star count

/* Return 0 if no active sponsors                                  */
if len(activeSponsors) == 0 then
    return 0;

/* Prepare query based on database type                            */
query ← ‘ SELECT COUNT(DISTINCT user_id) FROM sponsor_event WHERE
  repo_id = repoId AND user_id IN (activeSponsors) AND un_sponsor_at
  IS NULL‘;
```

MultiplierCount Implementation

The **Multiplier Sponsoring Count** metric calculates the number of unique Active User, to be recognised as activeSponsors in the following algorithms, who have acted as a foundation and set a multiplier on a specific repo.

Algorithm 13: GetMultiplierCount

```
Input: repoId: Repository ID, activeSponsors: Active Sponsor IDs
Output: int: Multiplier count

/* Return 0 if no active sponsors                                   */
if len(activeSponsors) == 0 then
    return 0;

/* Prepare query based on database type                             */
query ← ‘ SELECT COUNT(DISTINCT user_id) FROM multiplier_event WHERE
  repo_id = repoId AND user_id IN (activeSponsors) AND
  un_multiplier_at IS NULL‘;
```

ActiveFFSUserCount Implementation

The **Active FFS User Count** metric measures the number of users actively contributing to any repos over a defined period. It assesses engagement by tracking activity levels among contributors so that they can push their own repo.

Algorithm 14: GetActiveFFSUserCount

```
Input: repoId: Repository ID, activeUserMinMonths: Minimum activity period,
       latestRepoSponsoringMonths: Timeframe for sponsorships, isPostgres:
       Database type
Output: int: Amount of repos that all contibutors have sponsored

/* Retrieve associated emails for the repository                    */
emails ← GetRepoEmails(repoId);
/* Map emails to user IDs                                           */
userIds ← FindUsersByGitEmails(emails);
/* Filter active users based on activity threshold                  */
activeUsers ← FilterActiveUsers(userIds, activeUserMinMonths,
  isPostgres);

/* Count trusted repos for active users                             */
return len(activeUsers);
```

6.2.5 Thresholds and Scoring Distribution

The implementation of the Threshold is fairly straightforward. The initial values are inserted into the database. Future changes can be inserted into the database either by SQL commands or from the frontend. Naturally both actions require elevated permission.

Bound	Contributor Count	Commit Count	Sponsoring Count	Multiplier Count	Star Count	Active FFS User Count
Lower	4	40	1	5	1	1
Upper	13	130	10	20	5	10
Weight	2	2	2	2	1	1

Table 6.5: Health Value: Metrics upper and lower bounds

To get a better feel for how the thresholds will actually affect the HealthValue, following are 6 tables describing the PartialHealthValue point distribution, when the HealthValue calculation is triggered.

	Points
4	0.2
5	0.4
6	0.6
7	0.8
8	1.0
9	1.2
10	1.4
11	1.6
12	1.8
13	2.0

Table 6.6: Distribution Contributor Count

	Points
40	0.02
50	0.24
60	0.46
70	0.68
80	1.90
90	1.12
100	1.34
110	1.56
120	1.78
130	2.0

Table 6.7: Distribution Repo Commit Count

	Points
5	0.12
6	0.26
9	0.62
11	0.88
13	1.12
14	1.26
15	1.38
17	1.62
19	1.88
20	2.0

Table 6.8: Distribution Sponsoring Count

	Points
4	0.2
5	0.4
6	0.6
7	0.8
8	1.0
9	1.2
10	1.4
11	1.6
12	1.8
13	2.0

Table 6.9: Distribution Multiplier Count

	Points
1	0.2
2	0.4
3	0.6
4	0.8
5	1.0

Table 6.10: Distribution Star Count

	Points
1	0.1
2	0.2
3	0.3
4	0.4
5	0.5
6	0.6
7	0.7
8	0.8
9	0.9
10	1.0

Table 6.11: Distribution Active FFS User Count

6.2.6 Example of Health Value Application

Repository	Contributor Count	Commit Count	Sponsor Count	Multiplier Sponsoring Count	Star Count-donated	Active FFS User Count
Repo 1	32	268	4	8	5	1
Repo 2	0	0	0	10	1	0
Repo 3	5	47	0	15	0	4
Repo 4	4	19	0	0	1	7
Repo 5	53	176	6	0	1	0
Repo 6	0	0	0	0	3	2
Repo 7	0	0	8	23	1	0
Repo 8	10	17	9	11	1	4
Repo 9	13	73	1	2	2	5
Repo 10	36	625	2	1	0	0
Repo 11	2	3	8	5	1	9
Repo 12	14	219	12	8	1	0
Repo 13	3	4	1	7	0	8
Repo 14	0	0	8	9	2	7
Repo 15	45	317	22	14	0	0
Repo 16	0	0	0	19	0	2
Repo 17	27	200	9	17	1	0
Repo 18	29	2463	8	14	2	3
Repo 19	5	41	0	15	1	0

Table 6.12: Health Value Metrics Example

Repo	Contributo Count	Commit Count	Sponsor Count	Multiplier Sponsor-ing Count	Star Count	Active FFS User Count	Health Value
Repo 1	2	2	0.8	0.5	1	0.1	6.4
Repo 2	0	0	0	0.75	0.2	0	0.95
Repo 3	0.4	0.18	0	1.38	0	0.4	2.36
Repo 4	0.2	0	0	0	0.2	0.7	1.1
Repo 5	2	2	1.2	0	0.2	0	5.4
Repo 6	0	0	0	0	0.6	0.2	0.8
Repo 7	0	0	1.6	2	0.2	0	3.8
Repo 8	1.4	0	1.8	0.88	0.2	0.4	4.68
Repo 9	2	0.75	0.2	0	0.4	0.5	3.85
Repo 10	2	2	0.4	0	0	0	4.4
Repo 11	0	0	1.6	0.13	0.2	0.9	2.83
Repo 12	0	2	2	0.5	0.2	0	4.7
Repo 13	0	0	0.2	0.38	0	0.8	1.38
Repo 14	0	0	1.6	0.63	0.4	0.7	3.33
Repo 15	2	2	2	1.25	0	0	7.25
Repo 16	0	0	0	1.88	0	0.2	2.08
Repo 17	2	2	1.8	1.63	0.2	0	7.63
Repo 18	2	2	1.6	1.25	0.4	0.3	7.55
Repo 19	0.4	0.05	0	1.38	0.2	0	2.03

Table 6.13: Partial Health Values Example

Conclusion

All the data until now considered, we've decided on which thresholds to choose. Over the course of the lifetime of FlatFeeStack, the Health Value will see a higher significance in its internal values over the external values. This can also be enforced, by changing the weight a single metric can have.

In the initial inception phase of the multiplier the Health Value will be highly reliant on external metrics. But with increased interaction with FFS the internal metrics will increase in significance.

6.3 Fee Multiplier

Everything until now discussed are just supporting tools to the Fee Multiplier. The solution we strife is a multiplier that is applicable by foundations to any arbitrary repository. Combined with he approach of HealthValue and Verified Repos, we are sure to deliver a robust and fraud resistant method for fee

multiplication.
This calculation determines if and how the multiplier fee is distributed across repos and contributors while ensuring adherence to payout limits and fraud prevention mechanisms. Key steps in the calculation are:

- **Pool Calculation:** (See also getVerifiedMoneyPool)
  The total sponsorship amount is divided equally among all repositories. The share per repository (`amountPerRepo`) is computed by dividing the sponsor's total contribution (`SponsorAmount`) by the number of supported repositories, then multiplying this amount by the number of trusted repositories (`TrustedRepoSelected`).
- **Payout Limit:** A fraud protection mechanism ensures that users sponsoring themselves to manipulate payouts receive less than they contributed. The payout limit is calculated as:
  - For **current contributions**: `amountPerRepo * 0.9`.
  - For **Future Contributions**: `amountPerRepo`.
- **Deduction Logic:** The multiplier fee is then divided among the developers based on how much they have made, in the same way as for sponsorship. The only thing that is checked here is whether a foundation is in a position to pay based on the daily multiplier sponsorship limit and the funds that this foundation still has.

6.3.1 Algorithms for Fee Multiplier Calculation

The Fee Multiplier calculation is broken into distinct algorithms to streamline the process and ensure clarity. Each algorithm addresses a specific step in the overall distribution logic, from determining the pool to managing payout limits and distributing contributions effectively.

- **calcAndDeductFoundation:** This CalcAndDeductFoundation Algorithm is responsible for calculating the distribution pool (`pool`) and determining the Payout Limit (`payoutlimit`). It delegates the detailed deduction process to the `doDeductFoundation` algorithm.
- **doDeductFoundation:** DoDeductFoundation Algorithm handles the repository-level distribution by calculating the per-repository amount (`amountPerRepo`), applying fraud protection through the Payout Limit, and managing unclaimed funds or future contributions.
  `GetValidatedFoundationsSupportingRepo`, how its written in the code, are foundations that have still the capability to pay, so the daily multiplier sponsoring limit is not reached yet and they have still enough fonds.
- **handleFoundationPayout:** HandleFoundationPayout Algorithm ensures that contributions adhere to the daily multiplier sponsoring limit and available funds of a foundation, splitting contributions appropriately among verified and unverified contributors.
- **processUnclaimedFunds:** ProcessUnclaimedFunds Algorithm specifically handles cases where a contributor is not mapped, ensuring that unclaimed funds or future contributions are appropriately processed. This algorithm plays a crucial role in managing contributions for users who

are either unregistered or have not claimed their sponsorships. It also facilitates the handling of future contributions that are earmarked but not immediately distributed.

- **Current Contributions:** For current contributions, the algorithm divides the appropriate amounts for distribution among the contributors.
- **Future Contributions:** In cases of future contributions, the algorithm aims to record these contributions to maintain accurate financial tracking. This is particularly important to ensure transparency and so that in the future when there are contributors they get rightfully paid.
- **TODO and Proposed Fix:**
  * The algorithm contains a `TODO` to create a new database table specifically for tracking the history of future contributions. This table is intended to store unique fields, such as the foundation ID, repository ID, currency, and the day it was processed.
  * The absence of such a table currently limits the auditability and scalability of future contributions, as it may lead to overlaps or inconsistencies in financial records. Implementing this table would not only enhance the clarity of financial transactions but also improve compliance and reporting capabilities.
  * This fix would involve schema updates to the database, alongside updates to the algorithm to write to this new table whenever a future contribution is processed.

Algorithm 15: calcAndDeductFoundation

```
Input: sponsorDonations: Sponsorship data, parts: Divisions, yesterdayStart:
        Calculation date, futureContribution: Current or future
Output: Error: Any error during deduction

foreach block in sponsorDonations do
    if len(block.TrustedRepoSelected) > 0 then
        /* Calculate per-repo amount and trusted pool                    */
        allRepos ← block.TrustedRepoSelected +
         block.UntrustedRepoSelected;
        amountPerRepo ← block.SponsorAmount / len(allRepos);
        pool ← amountPerRepo * len(block.TrustedRepoSelected);
        /* Determine payout limit                                        */
        if futureContribution then
            payoutlimit ← amountPerRepo;
        else
            payoutlimit ← amountPerRepo * 0.9;
        /* Compute amount per part and delegate to deduction             */
        amountPerPart ← pool / parts;
        return doDeductFoundation(allRepos, yesterdayStart,
         block.Currency, amountPerPart, payoutlimit,
         futureContribution);
return nil;
```

Return to description

Algorithm 16: doDeductFoundation

```
Input: rids: Repository IDs, yesterdayStart: Day, currency: Currency type,
       amountPerPart: Share per division, payoutlimit: Fraud prevention limit,
       futureContribution: Current or future
Output: Error: Any error during processing

foreach rid in rids do
    /* Get contributor weights                                          */
    uidInMap, uidNotInMap, total ← getContributorWeights(rid);
    if uidInMap == nil and uidNotInMap == nil then
        continue;
    /* Retrieve supporting foundations                                  */
    foundations ← db.GetValidatedFoundationsSupportingRepo(rid,
     currency, yesterdayStart);
    foundationCount ← len(foundations);
    totalAmountForRepo ← foundationCount * amountPerPart;
    /* Apply payout limit                                               */
    if totalAmountForRepo > payoutlimit then
        amountFoundation ← payoutlimit / foundationCount;
    else
        amountFoundation ← amountPerPart;
    handleFoundationPayout(foundations, amountFoundation, currency,
     yesterdayStart, futureContribution);
return nil;
```

Return to description

Algorithm 17: handleFoundationPayout

```
Input: foundations: Supporting foundations, amountFoundation: Amount to distribute,
       currency: Currency type, yesterdayStart: Date, futureContribution: Current
       or future
Output: Error: Any error during processing

foreach foundation in foundations do
    /* Apply daily and fund availability checks                          */
    amountFoundationToPay ← CheckDailyLimit(foundation,
     amountFoundation);
    if amountFoundationToPay == -1 then
        continue;
    amountFoundationToPay ← CheckFondsAmount(foundation,
     amountFoundationToPay);
    if amountFoundationToPay == -1 then
        continue;
    /* Process unclaimed funds if needed                                  */
    processUnclaimedFunds(foundation, amountFoundationToPay,
     currency, yesterdayStart, futureContribution);
return nil;
```

Return to description

Algorithm 18: processUnclaimedFunds

```
Input: foundation: Foundation data, amount: Amount to process, currency: Currency
       type, yesterdayStart: Date, futureContribution: Current or future
Output: Error: Any error during processing

if futureContribution then
    /* TODO: Create a new table for tracking future contributions        */
    InsertOrUpdateFutureContribution(foundation, amount);
else
    /* Distribute or deduct amounts for current contributions            */
    DistributeOrDeduct(foundation, amount);
return nil;
```

Return to description

Chapter 7

UI Design

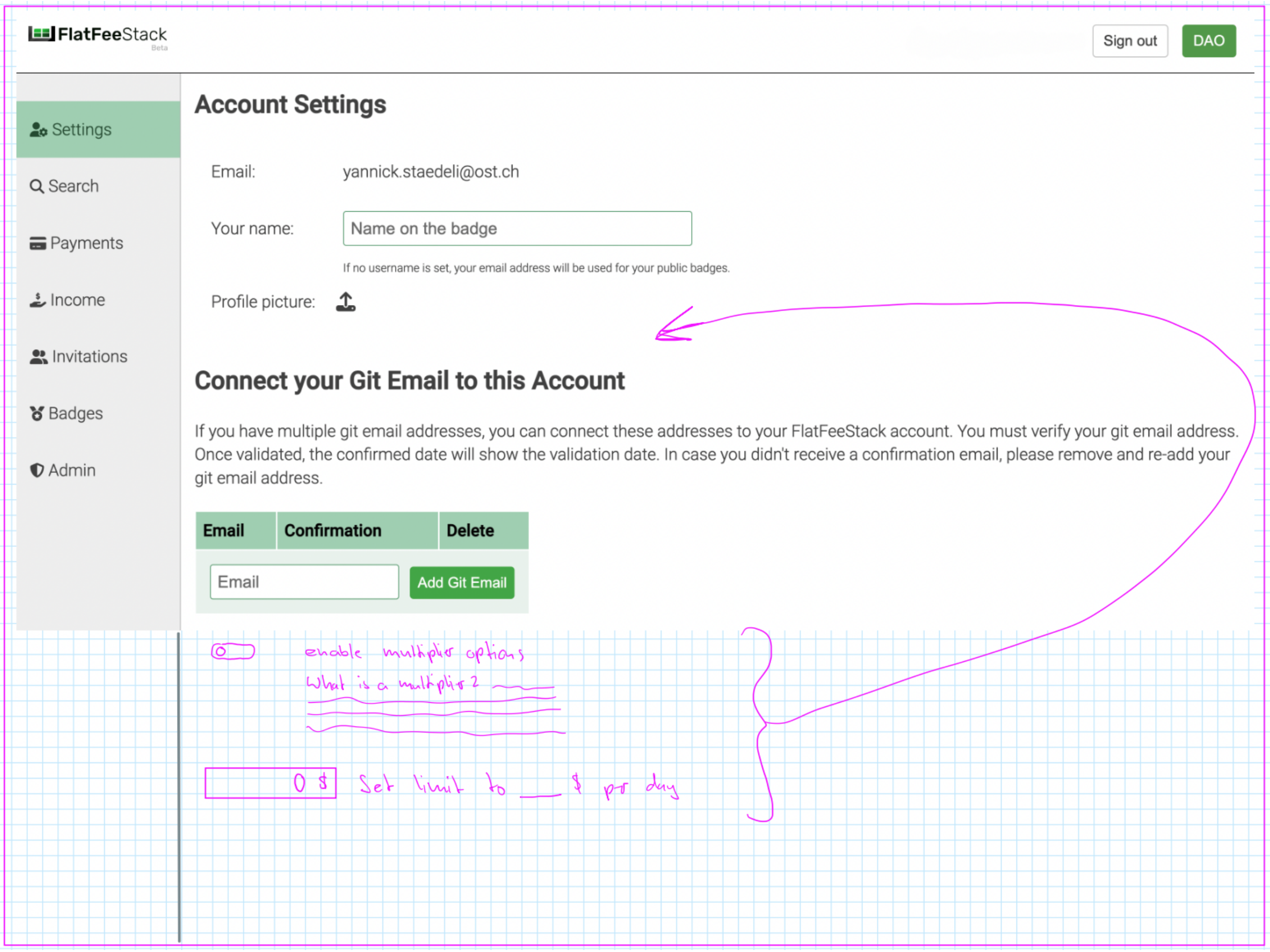
In the process of implementing new features on the FlatFeeStack platform, designing an intuitive and efficient user interface was a crucial step. While this was not a project built from the ground up, we adhered to established and proven UI design methodologies to ensure the frontend development aligned with our goals. Our approach followed a structured process: beginning with hand-drawn sketches to conceptualize ideas, advancing to detailed mockups for clarity and refinement, and then the actual programming of the frontend. This chapter outlines the key principles and practices of UI design we employed.

7.1 Sketches

We used OneNote on a tablet to draw the sketches and have them at any device accessible. With this approach everyone was able to make changes at the initial ideas.

Because we make additional changes to the existing project, we took a screenshot of the pages, that we are going to apply the new features. Each change is drawn in pink.

This were the initial sketches:



Account Settings

Email: yannick.staedeli@ost.ch

Your name: Name on the badge

If no username is set, your email address will be used for your public badges.

Profile picture:

Connect your Git Email to this Account

If you have multiple git email addresses, you can connect these addresses to your FlatFeeStack account. You must verify your git email address. Once validated, the confirmed date will show the validation date. In case you didn't receive a confirmation email, please remove and re-add your git email address.

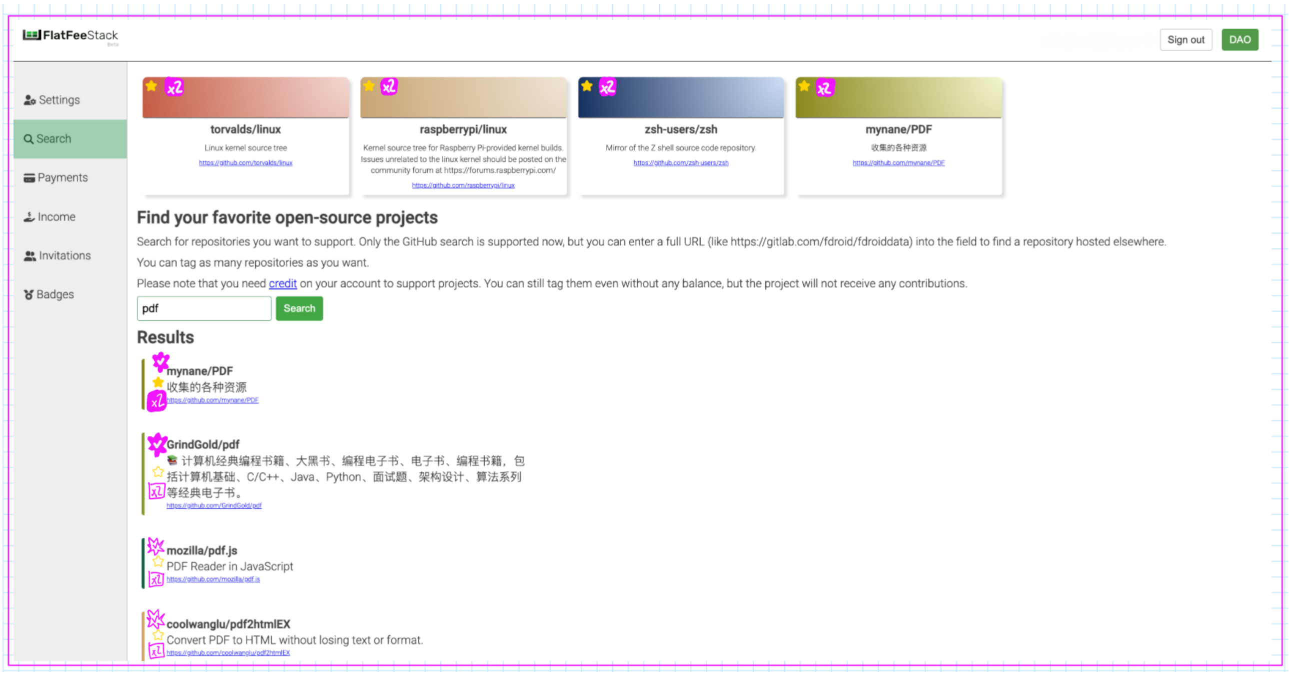
Email	Confirmation	Delete
Email	Add Git Email	

enable multiplier options

What is a multiplier?

0 $ Set limit to ___ $ per day

Figure 7.1: Sketch - Settings: Multiplier Options



torvalds/linux
Linux kernel source tree
https://github.com/torvalds/linux

raspberrypi/linux
Kernel source tree for Raspberry Pi-provided kernel builds. Issues unrelated to the linux kernel should be posted on the community forum at https://forums.raspberrypi.com/
https://github.com/raspberrypi/linux

zsh-users/zsh
Mirror of the Z shell source code repository.
https://github.com/zsh-users/zsh

mynane/PDF
收集的各种资源
https://github.com/mynane/PDF

Find your favorite open-source projects

Search for repositories you want to support. Only the GitHub search is supported now, but you can enter a full URL (like https://gitlab.com/fdroid/fdroiddata) into the field to find a repository hosted elsewhere.

You can tag as many repositories as you want.

Please note that you need credit on your account to support projects. You can still tag them even without any balance, but the project will not receive any contributions.

pdf Search

Results

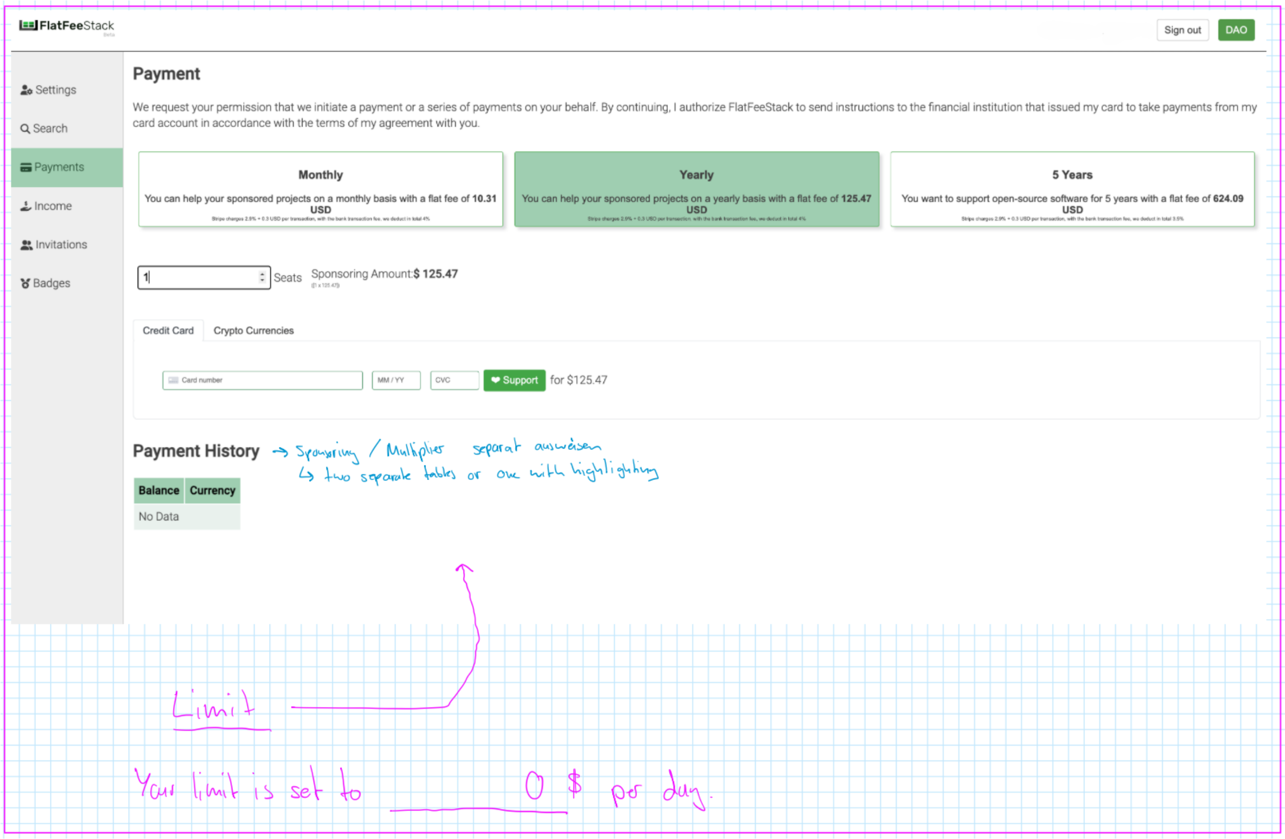
mynane/PDF
收集的各种资源
https://github.com/mynane/PDF

GrindGold/pdf
📚 计算机经典编程书籍、大黑书、编程电子书、电子书、编程书籍，包括计算机基础、C/C++、Java、Python、面试题、架构设计、算法系列等经典电子书。
https://github.com/GrindGold/pdf

mozilla/pdf.js
PDF Reader in JavaScript
https://github.com/mozilla/pdf.js

coolwanglu/pdf2htmlEX
Convert PDF to HTML without losing text or format.
https://github.com/coolwanglu/pdf2htmlEX

Figure 7.2: Sketch - Search: Multiplier Activation



Payment

We request your permission that we initiate a payment or a series of payments on your behalf. By continuing, I authorize FlatFeeStack to send instructions to the financial institution that issued my card to take payments from my card account in accordance with the terms of my agreement with you.

Monthly
You can help your sponsored projects on a monthly basis with a flat fee of **10.31 USD**
Stripe charges 2.9% + 0.3 USD per transaction, with the bank transaction fee, we deduct in total 4%

Yearly
You can help your sponsored projects on a yearly basis with a flat fee of **125.47 USD**
Stripe charges 2.9% + 0.3 USD per transaction, with the bank transaction fee, we deduct in total 4%

5 Years
You want to support open-source software for 5 years with a flat fee of **624.09 USD**
Stripe charges 2.9% + 0.3 USD per transaction, with the bank transaction fee, we deduct in total 3.5%

1 Seats Sponsoring Amount:**$ 125.47**
(1 x 125.47)

Payment History

Balance	Currency
No Data	

→ Sponsoring / Multiplier separat ausweisen
↳ two separate tables or one with highlighting
Limit
Your limit is set to 0 $ per day.

Figure 7.3: Sketch - Payments: Daily Limit

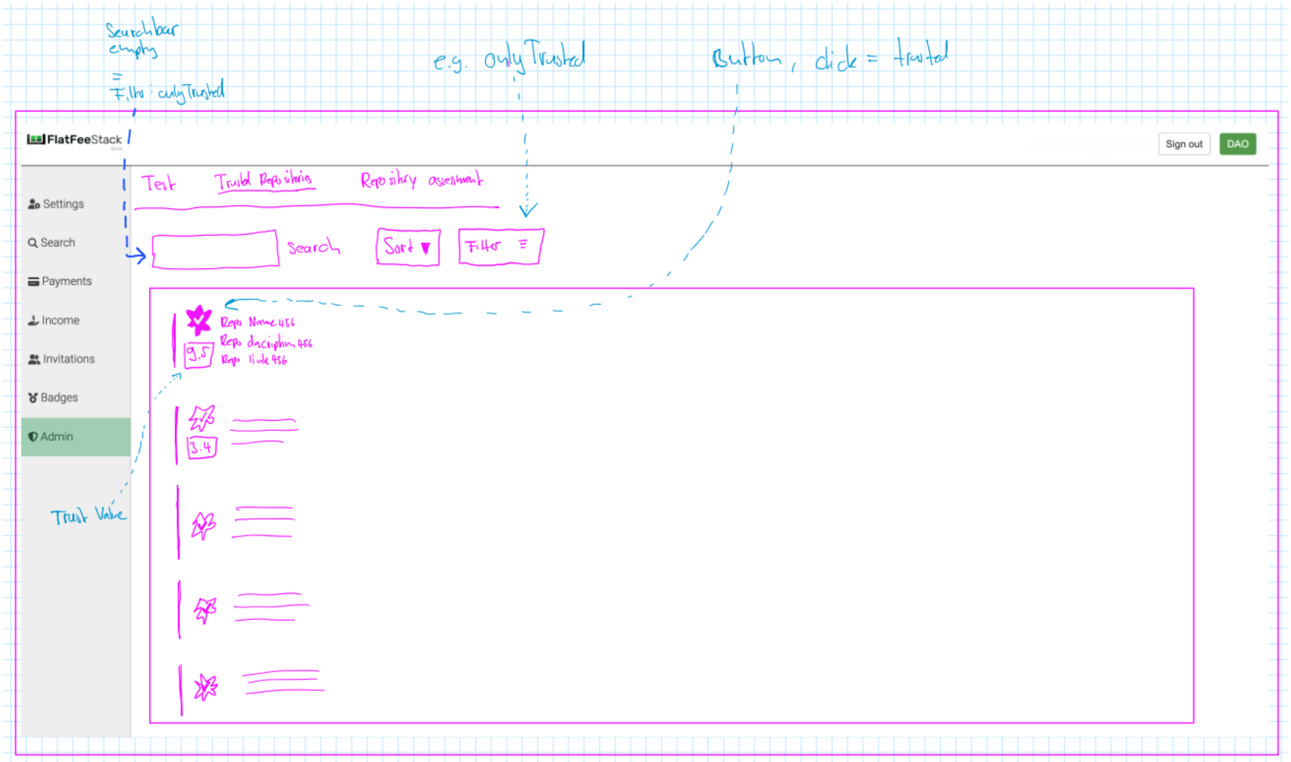


Figure 7.4: Sketch - Admin: Verified Repos

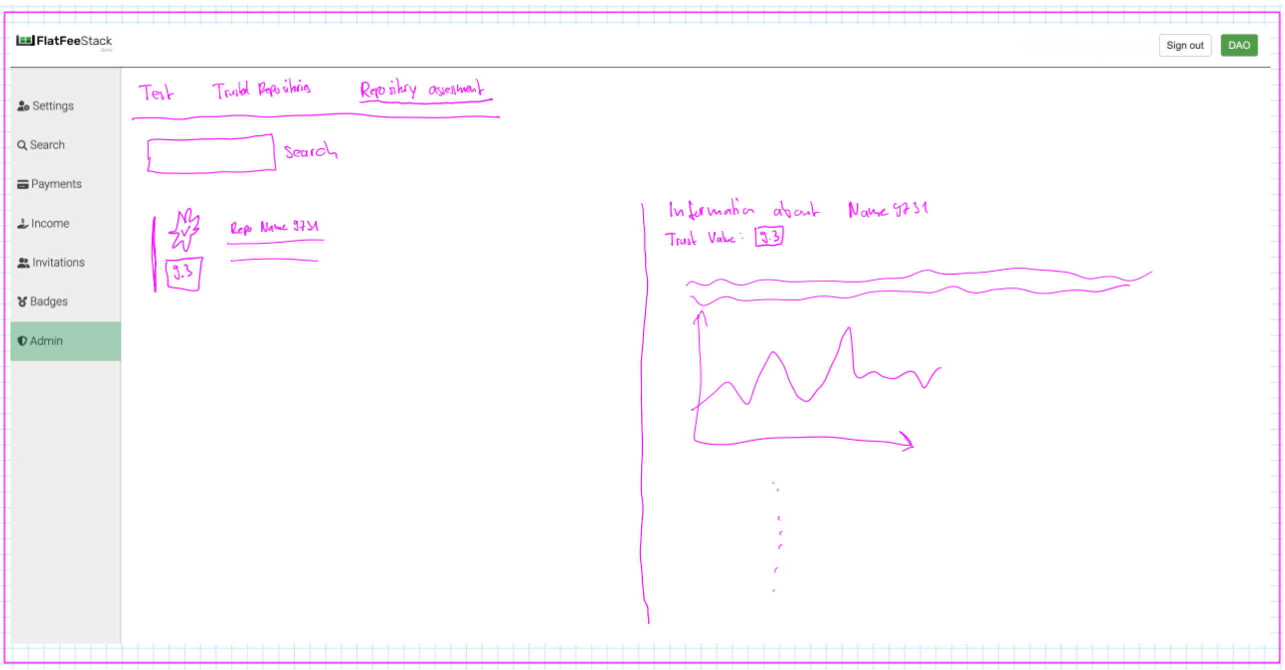


Figure 7.5: Sketch - Admin: Repo Assessment

7.2 Mockups

Following our collaborative efforts on sketching and discussing the website's new layout in the SA group and with our stakeholder Thomas Bocek, we utilized the online tool moqups to create the user interface mockup. This process of reviewing both sketches and the mockup within our team proved to be invaluable, enabling us to make critical decisions early on. Furthermore, this mockup will serve as a blueprint for the actual coding and development of the website, ensuring a cohesive and aligned implementation of the design. We tried to draw the mockups without colors, to make such design decision while developing the frontend. Our mockup is accessible via this link, but one must first create an account. After that, you can click on “Vorschau” / "Preview" in the top-right corner and navigate through the website yourself.

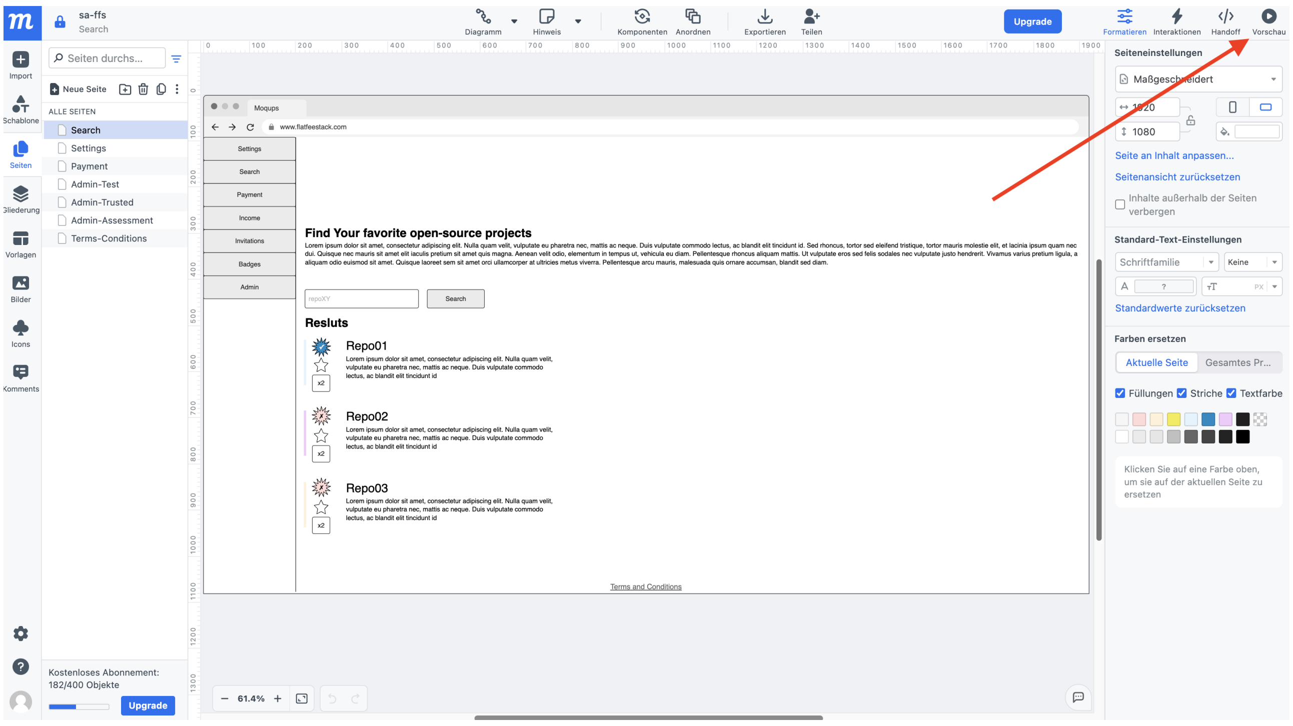


Figure 7.6: Mockup - Start Preview

All buttons are clickable and you can navigate through the application with the navigation bar.

7.3 Styling Decisions

All the design decision were made by the whole group. The leading frontend developer of a feature had to make an example, it then was discussed in the group or even with our stakeholder.
For most of the elements we took the color variables already defined in the global app.css.

7.3.1 Important Decisions

- **Navigation:** We decided to rename the existing Admin Page to "Test". We then created an additional admin page called "Healthy Repos". This represents an admin dashboard to handle everything about healthy repositories.
  The following functions are available on this page:
  - search through all available git repositories and add or remove them from our *Healthy Repos List*
  - sort the *Healthy Repos List* by date or score
  - search the *Healthy Repos List* by name
  - limit the amount of shown repos from the *Healthy Repos List*
  - with a click on the *Health Value* of a repo, show an overlay with the exact *Repo Assessment*
- **Fixed-Layout Website:** We decided to make these pages that we created with a fixed header, footer, and navigation bar. Furthermore, on the *Healthy Repos* page we also made the *Healthy*

Repos List fixed to further enhance the user experience. With this approach we aim to minimize mistakes that could occur while scrolling trough repos and accidentally marking a repo as *healthy*.

- **No Repo Assessment Page:** During development we changed our minds a few times on how to show the repo assessment. While our mockup shows a separate sub page, we decided to use the Healthy Repos page and add an overlay function for each repo.

7.4 Responsive Design

This is one of our limitations. The Responsive design is not fully implemented. It already had a low priority for our advisors. Most of the frontend pages we created or changed are only for admins, and we think that most of the admins will use a PC to use FlatFeeStack admin features.

Tested Screens

- MacBook Pro 16" (M3 Pro) built-in display
- Dell U2719 27" external display

7.5 Browser Compatibility

We tested our application on the most common web browsers. Our application is supported by all of them.

- Google Chrome
- Firefox
- Microsoft Edge
- Safari
- Brave

Chapter 8

Quality Measures

8.1 Development Procedure

This chapter outlines the tools and techniques we use to develop, test, and maintain our code and documentation. Given FlatFeeStack's development stage, it's crucial to document our existing tools and technology stack to prevent unnecessary complexity from introducing new technologies.

Beyond describing our technical infrastructure, this chapter establishes our testing framework and its associated Key Performance Indicators (KPIs). Testing remains a cornerstone of our development process, and we are committed to comprehensive test coverage for all features. While we don't currently implement automated metrics tracking, our developers can make informed decisions about their work's alignment with our established KPIs based on clear guidelines.

8.1.1 Code Tools

Diagnostics

Diagnostics delivers an incredible tooling for linting, syntax highlighting, automated indentation and more. For a development project this extension is very valuable. It automatically increases the quality and readability of our code.

GoLand (IDE)

GoLand is not a code tool but a whole IDE, it should still be mentioned. GoLand integrates built-in debugging, code refactoring, import completion, linting and many more features. As an IDE specifically designed for developing high quality go code, it provides a great development experience.

Prettify

Prettify is an opinionated code formatter that automatically standardizes code style across various programming languages and file types. It eliminates discussions about code formatting by enforcing a consistent set of rules, such as line breaks, indentation, and syntax, without requiring manual configuration. By integrating Prettier into development workflows, developers can automatically format their code with a single command or through editor plugins, ensuring clean, uniform code with minimal effort.1

Docker

Docker is a tool used to build are different components into small isolated containers. As it was already implemented in FlatFeeStack we've tailored our feature development to work with Docker. In general it enables development and distribution of applications for decentralized purposes.

8.2 Documentation Procedure

Everyone from this project will be involved in writing the documentation. This is where issues may arise if people have different ideas or standards. Hence this tightly written guideline should help with documentation.

Technology Stack

- Our repository for our documentation is GitLab. It is meant for having a distributed repository and CI/CD capabilities.
- For the documentation part itself we use LaTeX. Thus the documentation will be in plain-text and compiled to PDF with a LaTeX interpreter.
- In regards to spellchecking there is no restrictions. This documentation will implement aspell for some simple and rudimentary spellchecking. This process will also be implemented in Gitlab.
  - Mac: Aspell can be installed with homebrew.
  - Windows: The binary can be downloaded from the Internet.
- Online LLMs, such as ChatGPT and Claude

LLM Usage

Throughout the documentation process, we made moderate use of AI language models such as ChatGPT and Claude. These tools were primarily used to improve spelling and grammar. This approach helped us enhance the overall text quality by improving coherence and cohesion.[Ope24] [Cla24]

Documentation principles

- Written text shall be in English.
- Abbreviations are allowed where it makes sense, otherwise it has to be written fully.
- Some redundancy in documentation is allowed, but really only where it is required. Example: Describing a piece of code in two different chapters.

- The main branch is the working branch, changes will be compiled before pushing to to the repo.

Member participation

- Every project member is required to actively participate in documentation.
- Everyone has to respect the mentioned tech stack and rules.
- Regularly review the documentation, the style and point out key issues.

Documentation context

- The documentation is split in two parts
  - Product documentation: Describes the implemented features and development process.
  - Project documentation: Describes our guidelines and project journey.

8.2.1 Documentation and Development

The documentation is maintained separately from the core development project. Since this documentation is being created as part of our Term Project, merging it with FlatFeeStack's code base would be impractical. Key decisions and implementations are primarily detailed in the solution and implementation sections, while the actual code reflects the practical implementation.

To enhance both FlatFeeStack's code base and this documentation, we will use this documentation to address gaps in explaining FlatFeeStack's architecture and standards. The technical documentation can be developed independently and later integrated into FlatFeeStack to complement its existing coding structure and guidelines.

8.3 Scope of Testing

This section presents the testing strategy to ensure the quality and performance of FlatFeeStack.

Testing Activities

- **Acceptance Testing:** Verifies software functions against specified requirements. (Testing of Functional Requirements and Non-Functional Requirements)
- **Usability Testing:** Evaluate FlatFeeStack by testing it on users (e.g. user interaction and satisfaction).
- **Compatibility Testing:** Checks application behavior across different systems.

Test Types

- **Automated Regression Tests:** Ensures that the application still performs as expected after a change.
- **User Experience Tests:** Assesses interface usability.
- **Cross-browser and Cross-platform Tests:** Ensures consistent operation across different platforms.

Test Levels

We use the levels of testing which were presented in various courses.

- **Unit Testing:** Initial testing of code units.
- **Integration Testing:** Tests unit combinations for functionality and performance.
- **Acceptance Testing:** Validates complete application scenarios for end-users.

Test Environments

- **Development Environment:** The local testing environment that developers can use during the development process.
- **GitHub Pipeline:** For running automated tests after each commit we will use the GitHub pipeline.
- **Go Test Suit:** For subproject wide automated and integrated testing the go test suite is very valuable.

Tools and Technologies

- **Unit Testing:** Utilizing go test for GoLang.
- **Automated Testing:** Integration with GitHub CI/CD.

Roles and Responsibilities

Key roles defined:

- **Developers:** Perform unit and integration tests.
- **Tester:** Manages test environments and executes plans.

Testing Schedule

In general there are no testing schedule. As we work agile, our feature implementations are broken down into user stories, placing them into two weeks long scrum sprints. Thus no merges will be done into our man feature branch, as long as the whole test suite case is not successful. Testing is done on a regular basis and with every push to the git repository with Github Actions.

Many of the details in this subsection are based on the technical report [MHF+24].

8.4 Key Performance Indicators

8.4.1 KPI 1: Correct Multiplier Application

Goal: Ensure the multiplier is applied accurately based on the system's calculations without errors.
Target: 100% of transactions use the correct multiplier factor (1.9x at most).

8.4.2 KPI 2: Transaction Integrity

Goal: Ensure that all transactions involving the multiplier are secure and processed without tampering or interference.
Target: 100% of transactions (if all repos are really verified) are verified to be secure, with zero reported incidents of fraud.

8.4.3 KPI 3: Comprehensive Testing

Goal: Every added code is either part of a unit, integration or regression test. Usability testing is also a possibility for testing elements where automated testing is not practical. **Target**: Every added function in db is unit tested. Every added function in api is part of integration, regression or usability testing.

8.4.4 Definition of Done

The Multiplier Extension is considered done when the following criteria are met:

1. **Functional Requirements Completed:**
   - The multiplier calculation has been fully implemented, integrated, and functions according to the specified logic.
2. **Non-Functional Requirements:**
   - **Accuracy:** The system ensures that all multiplier calculations are accurate (six decimal places) with no margin for error.

- **Performance:** The system is designed to handle up to 1000 concurrent users, with 95% of transactions completed in under 500 milliseconds, even under peak load conditions.
- **Security:** All transactions processed by the multiplier system are secure, with no security breaches or tampering detected.

3. **Key Performance Indicators Completed:**
   - All specified KPIs must be successfully met for the project to be considered complete.
4. **Code Quality and Documentation:**
   - Code has been peer-reviewed, with no critical bugs or security vulnerabilities.
   - Unit and integration tests cover all relevant aspects of the multiplier, achieving at least 90% code coverage.
   - Documentation is complete, including:
     - Technical documentation for developers.
     - User guide explaining how the multiplier works and any important details for administrators.
5. **Testing and Verification:**
   - All tests (unit, integration, and performance) have passed in both development and production environments.
   - Non-functional requirements like accuracy and performance have been validated against the acceptance criteria.
6. **Stakeholder Sign-Off:**
   - Stakeholder has reviewed and approved the functionality and performance of the multiplier system.
   - Final sign-off has been obtained from both the development team and project manager.

8.5 Guidelines

8.5.1 Document Guidelines

Having document guidelines will help us maintain the integrity of the document even when multiple parties are working on it simultaneously. As such, we will push all updates directly to the main branch. This approach ensures that we remain updated at all times, providing constant visibility into the status of ongoing modifications, as opposed to having changes confined to a separate branch and only becoming visible when nearing completion.

Furthermore, we will implement very basic commit rules for the documentation, as extended guidelines could lead to an over-complicating of the process, resulting in fewer frequent commits. These commit rules will include a basic commit message, that is structured by the chapter name (where a change was made), followed by a very basic description - what was changed.

To enhance consistency, titles will be capitalized, and quick aspell spell checks will be conducted when adding content. These practices are expected to improve the editing process, ultimately minimizing the time spent on verification and correction.

In summary, adhering to the following guidelines will facilitate our objectives:

- Capitalization in titles
- aspell correction
- No git branches
- Simple commit rule: <chapter name>: <description of change made>

8.5.2 Code Guidelines

For new code we develop, we will implement clean code guidelines as defined in Robert C. Martin's book "Clean Code." This includes using meaningful names that reveal intention, creating small functions that focus on a single responsibility, and adding purposeful comments that explain intent or clarify complex sections while avoiding redundant or misleading documentation. [Mar09]

The structure of new files should follow basic rules for vertical and horizontal formatting. To maintain readability, we aim to limit file length to approximately 200 lines. Similarly, to prevent horizontal scrolling and improve code clarity, we target a maximum line length of 120 characters, with 80 characters being the preferred length.

For existing codas files, we will attempt to apply these clean code principles where practical, without undertaking major rewrites. With new feature additions and modifications to existing code, we'll gradually align with these standards while respecting the current architecture.

Beyond these general guidelines, we have specific standards for different aspects of the software. The frontend team will follow svelte coding guidelines for proper styling, while the backend team will adhere to official Go code conventions.

Effective communication is essential for maintaining clean, comprehensible code across all team members, as certain code sections may not be immediately clear to everyone. Through our weekly and biweekly meetings, we'll ensure adherence to guidelines and provide constructive feedback on any unclear code sections.

In summary, we will look to adhere to the following guidelines:

- Naming Conventions:
  - Use meaningful names reflecting intention
- Functions:

    - Employ small functions dedicated to a single purpose
    - We will adopt camel case for naming conventions

- Commentary:
    - Include meaningful comments clarifying intent or clarifying complex sections
    - Avoid redundant and misleading comments

- Structural Formatting:
    - Restrict line length to a maximum of 120 characters, ideally 80 characters

- Frontend Guidelines:
    - Adhere to the svelte guidelines for consistent frontend coding scheme as much as possible
    - Adopt the already implemented coding scheme
    - Extend code only when necessary

- Backend Guidelines:
    - Adhere to the official Go guide lines as much as possible
    - Adopt the already implemented coding scheme
    - Extend code only when necessary

8.5.3 Version-Control Guidelines

In our project, we will use the Trunk-Based Development approach to version control management. This strategy involves merging small, frequent updates directly into our main branch - the trunk. Our trunk in this case is SA-2024-YSMPRC which is always up-to-date with the main branch but still containing our stable progress.

The updates will take place as merges in the form of merge requests. As such, before merging the branch back onto the main trunk, we will have another team member go over the code and accept the merge first. This approach ensures we remain agile and efficient throughout the lifecycle of the project, while also minimizing errors. Once a release candidate has been developed after significant testing it will be merged into main.

Branches should be created as follows: <branch prefix>/<jira ticket number>-<concise branch name> The possible prefixes are:

- feature: used for developing new features
- bugfix: used to fix bugs in the code
- experimental: used for new ideas or prototypes that should not be part of a release

For example:

- feature/FFS-12-Multiplier-Admin-Button

Finally, we will use semantic commit messages to add coherence when pushing changes. The format looks like this: <type of change>: <description of change>
The possible types of change are:

- feat: a feature was added to the code or is in development
- fix: something in the code was fixed
- style: formatting, missing semicolons, etc. - no production code change
- refactor: refactoring production code, for example, renaming a variable
- test: adding missing tests, refactoring tests; no production code change

As for the description of change, it should be kept short but as informative as possible.

Chapter 9

Testing

9.1 Test Cases

We have determined 12 tests corresponding to the features we will have implemented on FlatFeeStack in the end. These features are essential not only to the core identity of our application but also to the perspective of end users, who might perceive FlatFeeStack as incomplete without them.

9.1.1 Test Cases for Non-Admin Users

Test Case User 1

When a user or foundation wants to make a multiplier sponsoring, he has to activate the features in the "Settings" tab. But if a user does not want to set multipliers, it is crucial to not confuse him with unnecessary buttons. Therefore it is important to test that only users who want to see multiplier options, can actually see them.

ID	TCU-1
Reference	Epic: Multiplier Selection
Test Case Description	We try to activate the multiplier feature, and check whether the new features are available or not.
Preconditions	User must have an FFS account and System is up.
Test-Steps	1. go to login Page 2. login with users credentials 3. navigate to "Search" 4. enter any arbitrary repository you know (e.g. flatfeestack) 5. multiplier sponsoring activation/deactivation must not be available 6. navigate to "Settings" 7. toggle the switch to "enable multiplier options" 8. navigate back to "Search" 9. search again for any repository 10. multiplier sponsoring activation/deactivation must be available
Expected Results	The user can't see the multiplier options until he activates it by himself in the "Settings" tab.
Status	Done

Table 9.1: Test Case User - 1

Test Case User 2

The multiplier sponsoring activation works as intended.

ID	TCU-2
Reference	Epic: Multiplier Selection
Test Case Description	We try to activate and deactivate the multiplier sponsoring.
Preconditions	User must have an FFS account and System is up.
Test-Steps	1. navigate to "Search" 2. activate multiplier sponsoring for any arbitrary and not already sponsored repo (e.g. flatfeestack) 3. the repo must appear in the sponsored repos, having sponsoring and multiplier sponsoring active 4. stop multiplier sponsoring the repo 5. the repo must still be marked as sponsored repo 6. activate again multiplier sponsoring 7. deactivate sponsoring for the repo 8. the repo must disappear from sponsored and multiplier sponsored repos
Expected Results	The user can successfully activate and deactivate multiplier options.
Status	Done

Table 9.2: Test Case User - 2

Test Case User 3

Users and foundations should have a clear and convenient overview to see where the multiplier sponsoring is active/inactive. Therefore they should be able to see it on first sight.

ID	TCU-3
Reference	Epic: Multiplier Selection
Test Case Description	We try to find out which repos only have a normal sponsoring and which have also a multiplier sponsoring.
Preconditions	User must have an FFS account and already a few repos he is sponsoring and multiplier sponsoring. System is up.
Test-Steps	1. go to login Page 2. login with users credentials 3. navigate to "Search" 4. take a look at the repo cards and distinguish between "sponsored repos" and "multiplier sponsored repos"
Expected Results	The user can quickly and successfully verify which repositories he is currently sponsoring and identify those with an additional multiplier sponsorship applied.
Status	Done

Table 9.3: Test Case User - 3

Test Case User 4

Users and foundations understand which repositories are safe to make multiplier sponsoring.

ID	TCU-4
Reference	Epic: Fraud protection blackbox
Test Case Description	We try to find out which repos are assessed as healthy by the admins of the platform.
Preconditions	User must have an FFS account. System is up.
Test-Steps	1. navigate to "Search" 2. take a look at the search results and identify the repos that are most likely legitimate
Expected Results	The user can easily check which repos from Flatfeestack are marked verified and which are not
Status	Done

Table 9.4: Test Case User - 3

9.1.2 Test Cases for Admin Users

Test Case Admin 1

FlatFeeStack administrators should be able to manually mark a repository as healthy in the admin search results. To mark a repo as unhealthy should be possible in admin search results as well as in the overview cards To do so they must have the ability to search through all existing git repos based on their names.

ID	TCA-1
Reference	Epic: Healthy repo selection
Test Case Description	We try to mark an unhealthy repo as healthy and vice versa.
Preconditions	User must have an Admin FFS account. System is up.
Test-Steps	1. navigate to "Healthy Repos" 2. write into the "Search All Repos" input field the name of any repo (e.g. flatfeestack) 3. mark two of these search results as healthy 4. the marked repos must appear in the dashboard overview as separate cards and have a healthy icon 5. mark one of the two repos as unhealthy in the search results and wait at least 5 seconds 6. the repo now must have been disappeared from the dashboard overview and have the unhealthy icon 7. mark the other one of the two repos as unhealthy in the dash-board overview card and wait at least 5 seconds 8. the second repo must have been disappeared from the dash-board overview and have the unhealthy icon
Expected Results	The admin is able to successfully mark a repo as healthy or un-healthy.
Status	Done

Table 9.5: Test Case Admin - 1

Test Case Admin 2

FlatFeeStack administrators should be able to abort the process of marking a repo as unhealthy. But, the undo mechanism should not interfere with any other action on the platform such as change the page, search for another repo or simultaneously mark another repo as unhealthy.

ID	TCA-2
Reference	Epic: Healthy repo selection
Test Case Description	We try to mark a healthy repo and then immediately do something else.
Preconditions	User must have an Admin FFS account. System is up. Some testing repos already marked as healthy
Test-Steps	1. go to login Page 2. login with admin credentials 3. navigate to "Healthy Repos" 4. mark one repo as unhealthy and abort the change within 5 seconds 5. the repo must stay marked as healthy 6. mark one repo as unhealthy and click any other page and come back to "Healthy Repos" within 5 seconds 7. the repo must already be marked 8. the repo now must have disappeared from the dashboard overview
Expected Results	The admin is able to successfully mark a repo as healthy or unhealthy.
Status	Done

Table 9.6: Test Case Admin - 2

Test Case Admin 3

When a FlatFeeStack administrators manually marks a repository as healthy or unhealthy, it is visible for all other users on the platform.

ID	TCA-3
Reference	Epic: Healthy repo selection
Test Case Description	We try to mark an unhealthy repo as healthy and check if user sees it.
Preconditions	User must have an Admin FFS account. Another user account. System is up.
Test-Steps	**Admin Perspective** 1. navigate to "Healthy Repos" 2. mark one repo as healthy (e.g. flatfeestack) 3. navigate to "Search" 4. the repo now must have a healthy repo icon **User Perspective** 1. login with the user account 2. navigate to "Search" 3. the repo must have a healthy repo icon
Expected Results	The admin is able to successfully mark a repo as healthy that everyone sees it in "Search" tab.
Status	Done

Table 9.7: Test Case Admin - 3

Test Case Admin 4

A FlatFeeStack administrators should see at all time the precise and most current health value.

ID	TCA-4
Reference	Epic: Healthy repo selection
Test Case Description	We try to display the health value of each repo.
Preconditions	User must have an Admin FFS account. System is up.
Test-Steps	1. navigate to "Healthy Repos" 2. search for any healthy repo 3. health value in search part must match the value in repo card
Expected Results	The admin is able to successfully verify that the health value consistent over the whole platform.
Status	Done

Table 9.8: Test Case Admin - 4

Test Case Admin 5

A FlatFeeStack administrators should have a convenient dashboard about the healthy repos. It should be possible for him to see which healthy repos were recently or first added. And he should also be able to sort the healthy repos by health score.

ID	TCA-5
Reference	Epic: Healthy repo selection
Test Case Description	We try to find the healthy repo with the worst health score and the healthy repo that was first added.
Preconditions	User must have an Admin FFS account. System is up.
Test-Steps	1. navigate to "Healthy Repos" 2. identify the requested repos (sort function should be used)
Expected Results	The admin is able to successfully identify the repo withe lowest health value and the repo first added as healthy.
Status	Done

Table 9.9: Test Case Admin - 5

Test Case Admin 6

A FlatFeeStack administrators should see how and with which metrics the health value was calculated. He should gets a concise insight if he wants to know the exact repo metrics.

ID	TCA-6
Reference	Epic: Fraud protection blackbox
Test Case Description	We try to find out which metrics were used to calculate the health value.
Preconditions	User must have an Admin FFS account. System is up.
Test-Steps	1. navigate to "Healthy Repos" 2. identify the used thresholds to calculate the health value
Expected Results	The admin is able to successfully identify the thresholds used.
Status	Done

Table 9.10: Test Case Admin - 6

Test Case Admin 7

A FlatFeeStack administrators should get a concise insight if he wants to know the partial health values.

ID	TCA-7
Reference	Epic: Fraud protection blackbox
Test Case Description	We try to identify where a repo got how many points in the assessment (partial health values).
Preconditions	User must have an Admin FFS account. System is up.
Test-Steps	1. navigate to "Healthy Repos" 2. search for any arbitrary repo (e.g. flatfeestack) 3. identify how many points this repo got for which metrics
Expected Results	The admin is able to successfully identify how the system calculated the health value for one repo.
Status	Done

Table 9.11: Test Case Admin - 7

Test Case Admin 8

A FlatFeeStack administrators should have the ability to change the thresholds used to calculate Health Value.

ID	TCA-8
Reference	Epic: Fraud protection blackbox
Test Case Description	We try to change the threshold of the metric "Contributor Count" and "Commit Count".
Preconditions	User must have an Admin FFS account. System is up.
Test-Steps	1. navigate to "Healthy Repos" 2. find the threshold settings 3. change the threshold values for the metric "Contributor Count" and "Commit Count" 4. take a look at any repo assessment and verify your changes
Expected Results	The admin is able to successfully identify how the system calculated the health value for one repo.
Status	Done

Table 9.12: Test Case Admin - 8

9.2 Unit Tests

Every Developer implements and does unit testing where it makes sense. As the code base has a significant amount of test cases, we will adopt a similar style of testing. The developers are encouraged to use the principles of TDD, meaning to write the Unit Tests in advance to ensure a form of "safety net" and immediate feedback whether the code's behavior is unexpected. This approach also ensures there is no regression, where new features break previously existing functionality.

The use of Unit Tests will mainly occur in the backend and business side of the application, since Unit Testing the frontend would not make sense in this particular case. The frontend will be tested using Acceptance and Usability Tests.

Side note: Every developer is informed that using TDD isn't a mandatory approach, but more of a suggestion. If by chance the developer decides that writing Unit Tests in advance for a specific feature would be disadvantageous, he is free to approach the development as he pleases.

9.3 Usability Testing

After having implemented all necessary Functional Requirements, we will conduct a usability test. This test will be conducted by volunteers who have not been involved in the development of the application. The goal of this test is to identify any usability issues that may have been overlooked during the development process. The test will involve the volunteers performing a series of tasks using the application, while providing feedback on their experience. The feedback will be used to make any necessary improvements to the application before the development ends.

9.4 Test Plan

All participants will do the test in person and assume the role of one of the personas. The test will be done in a controlled environment, where the participants will be given a set of tasks to complete. The participants will be asked to think aloud while performing the tasks, so that the facilitator can observe their thought process and identify any usability issues. The feedback from the participants will be noted down and later be used to improve the UX/UI.

The purpose of this test plan is to validate the functionality, and reliability of the newly implemented features before they are released to production.

The testing scope includes all new features implemented in the application, covering functional, integration, acceptance and usability / end-user testing.

9.4.1 Acceptance Test Plan

The purpose of this subsection is to outline the test plan for the Acceptance Test, a crucial phase in the development process where the implemented features are verified against the agreed-upon requirements. The acceptance tests will primarily be conducted by the development team (acting as QA team) to ensure that all functionalities align with the specifications. Additionally, the main stakeholder, represented by our professor Thomas Bocek, will independently perform these tests. This approach ensures validation by the developers and external approval by the customer, providing a comprehensive evaluation of the system's readiness for deployment.

Template for Acceptance Test

Participant's Name, Job / Degree	
Browser / Device	TBD
Date	December 5, 2024
Location	OST Rapperswil
Tasks	FR1: Select Git Project → Participant Feedback: TBD FR2: View Multiplier Sponsoring History → Participant Feedback: TBD FR3: Toggle Multiplier → Participant Feedback: TBD FR4: Apply Multiplier → Participant Feedback: TBD FR5: View Multiplier Status → Participant Feedback: TBD FR6: Daily Multiplier Sponsoring Limit → Participant Feedback: TBD NFR1: Multiplier Accuracy → Participant Feedback: TBD NFR2: Performance under Concurrent Use → Participant Feedback: TBD NFR3: Dynamic Multiplier Calculation → Participant Feedback: TBD NFR4: Secure Multiplier Access → Participant Feedback: TBD
Overall Feedback	TBD
Status	TBD

Table 9.13: Template for Acceptance Test Protocol

All the detailed test protocols created during our testing meetings with participants are included in the appendix section. Test Reports

9.4.2 Acceptance Test Conclusion

Our initial plan was, to be finished with all programming work at least 1 week before the final hand-in. Therefore it would have been perfect to make all the tests 2 weeks before final hand-in.

We conducted our acceptance tests 8 days before hand-in. And as expected, we found some stuff that has to be done and takes e few hours to add.

Table 9.14: Acceptance Test Conclusion Critics

Fix or Limitation	**Critic**	**Solution (idea)**
Fix	The Payment History tables have no column to see when each entry was made.	Enhance table with this column, and provide information from backend.
Limitation	There is no option to test the multiplier with fake data in the frontend.	Adjust frontend and backend with according functionalities. The goals would be: (1) create new fake foundations (2) add multiplier sponsoring to any repo (3) fake sponsoring to the chosen repo
Fix	When a foundation cannot cover the daily multiplier sponsoring limit with their current balance, the payment is aborted. But in FR6 stipulates to apply a pay factor.	Adjust backend function accordingly.
Fix	The text above the daily multiplier sponsoring limit input field should be eliminated, it is not necessary like this.	The value that is already bound to the input field should be shown, according to the username input field.
Limitation	It is not convenient to have two fields where a foundation can / have to add the credit card information. Now is on Payment - and Multiplier Payment tab.	In the current box with the tabs "Credit Card" and "Crypto Currencies" only have the option to add the payment method. Then, beneath this box implement the button to actually trigger the donation. Respectively on the Multiplier Payment tab fill up your balance for multiplier sponsoring.
Limitation	It is not clear what the daily limit is when you pay with crypto currencies and have to set the limit in dollar.	If Crypto currency is enabled show the exchange rate, what the amount of dollar is in the crypto currency. Also update the rate daily for more accuracy.

Fix or Limitation	Critic	Solution (idea)
Limitation	NFR2 is not met entirely, performance is not tested.	Has very low priority, but could be performed when enough time is left. A Raspberry Pi 5 is needed.
Fix	The text of NFR3 is not accurate anymore.	Update the documentation.
Fix	Repo Assessment last metric is displayed wrongly as undefined.	Match the output handling to return 0 of no metric or value is present.

9.4.3 Usability Test Plan

This subsection outlines the plan for testing the system with end users. Although Usability and End-User Testing is not the same thing we perform these tasks in one. The tests will be conducted with individuals experienced in using computer systems to gather feedback on usability, functionality, and overall user experience. Their input will help identify any practical issues and ensure the system meets real-world expectations.

We separate the test groups into admins and foundations, where we either use the admin test cases or the user and foundation test cases.

Instruction for Tester: The participant will not have access to the test cases directly. You should read the test case introduction aloud to them, which is provided at the top of the tables. The test steps outline the actions the participant needs to perform. If the participant gets stuck, guide them using the instructions in the test steps.

Template for Test Group Foundations

- Number of Participants: 3

Participant’s Name, Job / Degree	
Browser / Device	TBD
Date	December 5, 2024
Location	OST Rapperswil
Facilitator	Yannick Städeli
Tasks	Test Case User 1 $\rightarrow$ Observations: TBD $\rightarrow$ Participant Feedback: TBD Test Case User 2 $\rightarrow$ Observations: TBD $\rightarrow$ Participant Feedback: TBD Test Case User 3 $\rightarrow$ Observations: TBD $\rightarrow$ Participant Feedback: TBD Test Case User 4 $\rightarrow$ Observations: TBD $\rightarrow$ Participant Feedback: TBD
Expected Results	Defined in Test Cases
Actual Results	TBD
Overall Feedback	TBD
Status	TBD

Table 9.15: Template for Usability Test Protocol with Foundation

Template for Test Group Admins

- Number of Participants: 4

Participant's Name, Job / Degree	
Browser / Device	TBD
Date	December 5, 2024
Location	OST Rapperswil
Facilitator	Yannick Städeli
Tasks	Test Case Admin 1 → Observations: TBD → Participant Feedback: TBD Test Case Admin 2 → Observations: TBD → Participant Feedback: TBD Test Case Admin 3 → Observations: TBD → Participant Feedback: TBD Test Case Admin 4 → Observations: TBD → Participant Feedback: TBD Test Case Admin 5 → Observations: TBD → Participant Feedback: TBD Test Case Admin 6 → Observations: TBD → Participant Feedback: TBD Test Case Admin 7 → Observations: TBD → Participant Feedback: TBD Test Case Admin 8 → Observations: TBD → Participant Feedback: TBD
Expected Results	Defined in Test Cases
Actual Results	TBD
Overall Feedback	TBD
Status	TBD

Table 9.16: Template for Usability Test Protocol with FFS Admin

9.4.4 Usability Test Conclusion

Conclusion Users

Although there were only three test subjects, they were very distracted. They all work with similar applications on a daily basis, but were not yet familiar with FlatFeeStack. A few wishes were expressed as to how FlatFeeStack could be improved in their eyes. These need to be validated and implemented if necessary.

Critics:

- The daily limit should also be set by entering an amount and focus out of the input, instead of enter or button. This would then be like changing the name
- If a willing foundation wants to select many repos, it stacks up and gets chaotic and enlarges the page vertically extremely
- Declare somewhere what exactly a healthy repo is and how it came about
- Show where the relevant options for a foundation are located when registering

Conclusion Admins

End-User testing is always very valuable to gain a different view on the application. We got some critics, and have seen some UI flaws that have to be taken care of.

Critics:

- Not clear that the repo assessment can be opened by clicking on the score. (Only by first usage)
- Not clear what each metric does when clicking on threshold settings.
- Dashboard of the healthy repos shows not enough repos, it should show more rows of repo cards. Another idea mentioned was, to toggle between list view and card view. With the list view you would see more repos at once.
- Get the ability to verify more than one repo simultaneously. E.g. add all repos from user "Microsoft/" as healthy.

UI Flaws to change:

- Multiplier features: (1) Icon is missing (2) No more information on "here" button
  - Fixed

Chapter 10

Conclusion and Prospect

The final chapter provides a summary of the project, focusing on the addressed problem, implemented solutions, and future improvements. Key features such as the Fee Multiplier Calculation, Health Scoring System, and Verification Controls are reviewed to highlight their technical contributions. The chapter also explores potential advancements, including repository health metrics, frontend enhancements, and testing improvements. Finally, lessons learned are discussed.

10.1 Problem

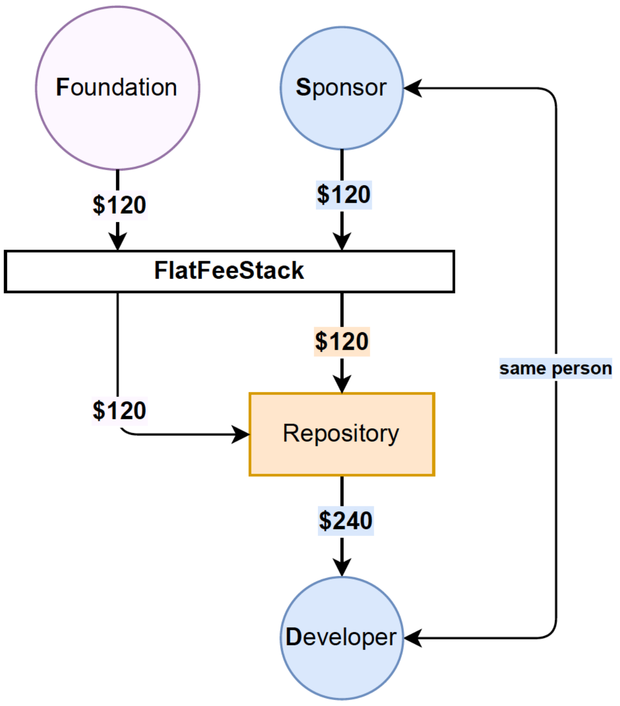


Figure 10.1: Developer exploiting weak Fee Multiplier feature

At first glance, the multiplier system sounds brilliant — until you hit a major snag. Imagine this: the sponsor is also the repository owner and the sole developer contributing to the project. Suddenly, they're receiving double the amount they paid in sponsorships. Congratulations, we've just invented an infinite money glitch! Every proposed fix seemed to bring its own complications. For instance, what if the multiplier only applies when a sponsor donates to two repositories? Sure, the payment gets split, but doubling half still gives the same amount as the original payment. Problem solved? Not quite. What if the developer owns both repositories? Same glitch, new disguise. Each solution we brainstormed felt brilliant — until the next thought unraveled it. It was like playing whack-a-mole with logic. At one point, this problem felt downright unsolvable. But as we Swiss like to say, "Think beyond the edge of the plate!" So, we re-engineered the multiplier concept and devised a solution that's not just innovative but also security-aware.

10.2 Solution

Our Fee Multiplier Solution is a robust framework designed to balance donation amplification with fraud protection. It consists of three critical components: a carefully designed **Fee Multiplier Calculation and Distribution** mechanism, ensuring fairness and security in sponsorship allocation; a **Verified/Unverified System** to categorize repositories based on credibility; and a **Health Scoring System** to evaluate repository quality using well-defined metrics.

10.2.1 Fee Multiplier Calculation and Distribution

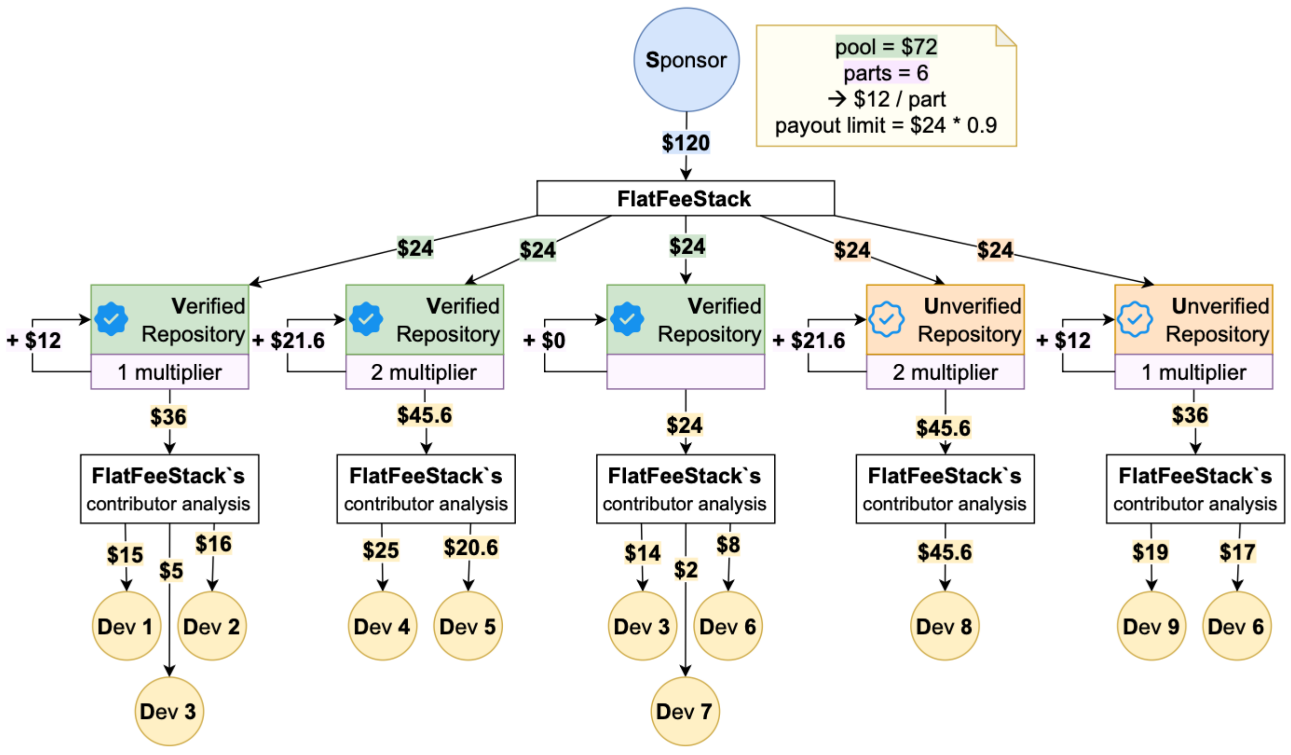


Figure 10.2: Key Achievement Example

This component employs an algorithmic approach to determine how multiplier sponsorship is distributed while safeguarding against abuse. Key features include:

- A daily multiplier limit to cap maximum amplified donations.

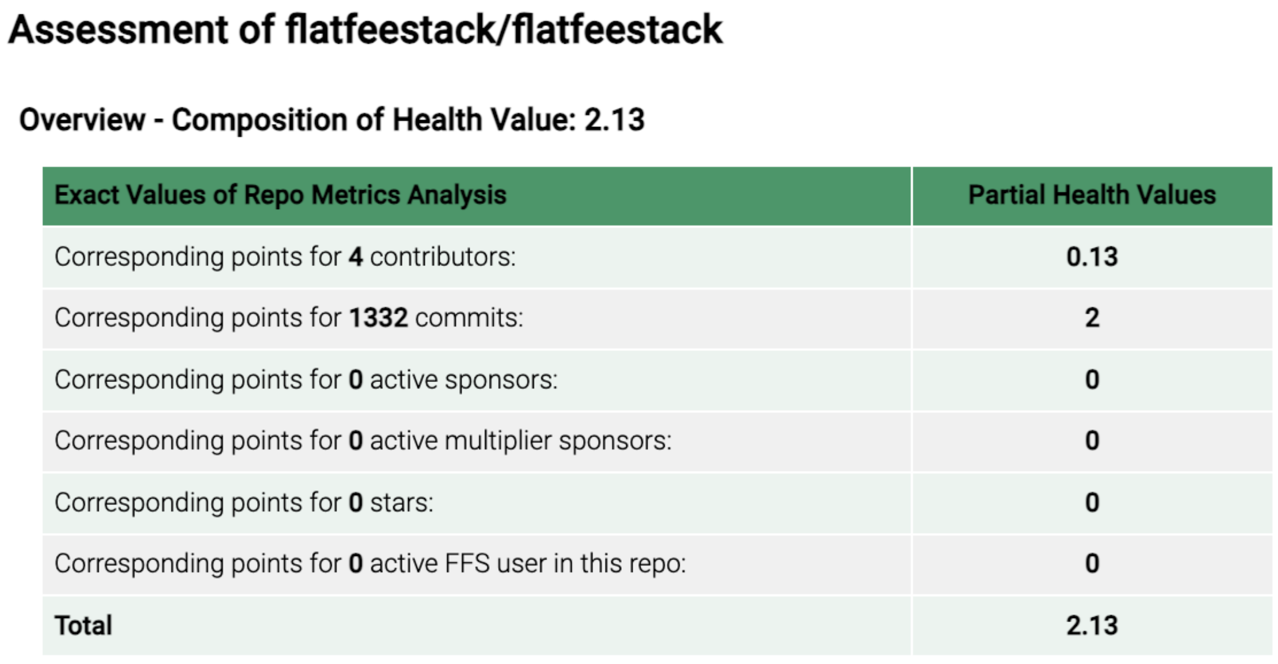
- Eligibility criteria for repositories to ensure funds are directed only to legitimate projects.

The sponsorship flow follows a structured process, visually represented in the diagram obove, with the following rules:

1. **Untrusted Repositories Only:** If a sponsor exclusively funds unverified repositories, no multiplier sponsorship will be applied by the foundation, even if it is set. This information is not visible by the foundation, they only see to which repos they have paid a multiplier fee.
2. **Trusted Repositories Only:** If there is at least one verified repository in the sponsorship bundle of the sponsor, each repository that has a multiplier on top, receives double the designated sponsorship amount, albeit with a 0.9 multiplier factor to regulate amplification.
3. **Mixed Sponsoring Bundles:** When trusted and untrusted repositories are combined, the system considers:
    - The amount of money that flows to trusted repositories as the **trusted money pool**.
    - The total multiplier amount in the bundle as parts of a **funding pool** also called **parts** in the example above.
    - The value of each part, calculated as the pool divided by the number of parts.

    Both trusted and untrusted repositories receive funds according to the number of multipliers assigned to them, ensuring fair distribution of "trusted money."

10.2.2 Health Scoring System: Enhancing Credibility



Assessment of flatfeestack/flatfeestack

Overview - Composition of Health Value: 2.13

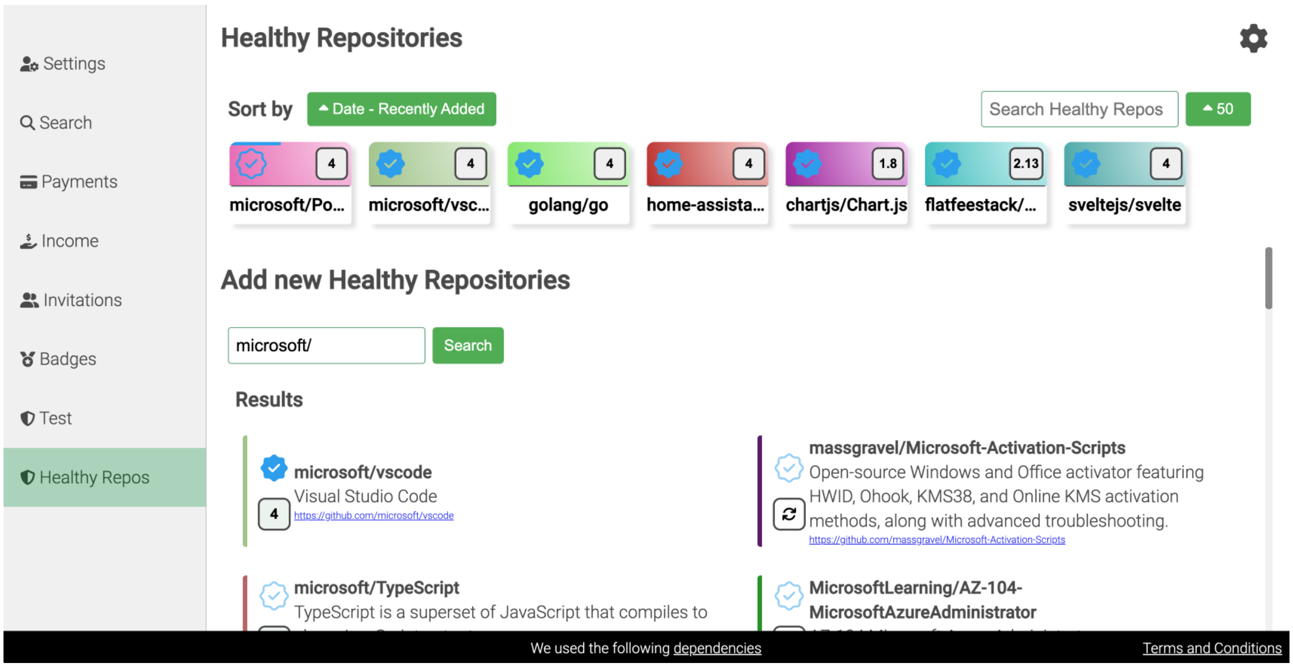
Exact Values of Repo Metrics Analysis	Partial Health Values
Corresponding points for **4** contributors:	**0.13**
Corresponding points for **1332** commits:	**2**
Corresponding points for **0** active sponsors:	**0**
Corresponding points for **0** active multiplier sponsors:	**0**
Corresponding points for **0** stars:	**0**
Corresponding points for **0** active FFS user in this repo:	**0**
Total	**2.13**

Figure 10.3: Repo assessment table

The **Health Scoring System** evaluates repositories using six metrics — two external and four internal factors — to generate a repository health value on a scale of 0 to 10, where StarCount and ActiveFFSUserCount give a maximum of 1 point and the remaining metrics 2 points each. This system offers:

- A transparent and objective assessment of repository quality.
- A critical layer of fraud protection by assigning a clear health score, ensuring only trustworthy repositories will be verified by admins.

10.2.3 Verification and Admin Controls



microsoft/vscode
Visual Studio Code
4
https://github.com/microsoft/vscode

massgravel/Microsoft-Activation-Scripts
Open-source Windows and Office activator featuring HWID, Ohook, KMS38, and Online KMS activation methods, along with advanced troubleshooting.
https://github.com/massgravel/Microsoft-Activation-Scripts

microsoft/TypeScript
TypeScript is a superset of JavaScript that compiles to

MicrosoftLearning/AZ-104-MicrosoftAzureAdministrator

Figure 10.4: Healthy repos page

Given the critical role of repository verification, we introduced an enhanced admin console to strengthen the trustworthiness of the platform. This admin interface allows:

- If a specific repository has not yet been analyzed, it is possible to click on the 'refresh' button (located where the score is displayed) to trigger the analysis.
- Perform a detailed inspection of repositories by clicking on the score of a specific repository. This opens an overlay with all information about the metrics.
- Search or sort all verified repositories and enable the ability to remove their verified status if necessary.
- Adjust the thresholds by accessing the settings menu in the top right corner.
- Ensure precise categorization of repositories as either verified or unverified.
- If there are too many verified repositories, it is possible to cap their number at a defined limit, or leave it uncapped and enable horizontal scrolling for easier navigation.

This ensures that **FlatFeeStack** maintains high standards of reliability and security across the board.

10.2.4 Foundation View - Daily Limit Input Mechanism

Lastly, while project methodology typically calls for a feature freeze in the final stages of development, we identified a minor, easily implementable improvement to the input mechanism for daily limits. The current approach required an explicit enter or button press to update the limit, which we simplified to an automatic update when the input field changes (on change). This modification mirrors the interaction pattern used for name changes, creating a more intuitive and streamlined user experience. As a quick fix that could be rapidly tested without risking project stability, it serves as a reminder that ideally, such improvements should be integrated earlier in the development cycle to maintain project momentum and prevent scope creep.

10.3 Term Project Closure

All Functional Requirements have been successfully fulfilled and implemented. Regarding the Non-Functional Requirements, all except one have been successfully implemented. The unfulfilled NFR is NFR2 (Performance under Concurrent Use), which could not be completed due to time constraints.

10.4 Prospect

This section provides a detailed overview of potential improvements to the features developed during this term project, as well as enhancements to the FlatFeeStack system as a whole.

10.4.1 Healthy Repos

Automated Verification Threshold

- Leverage existing metrics and health value calculations to establish a fixed point threshold
- Eliminate manual intervention in the verification process
- Automatically classify repositories based on predefined health criteria
- Audit additional security measures to prevent fraudulent verification

Bulk Verification Capability

- Introduce ability to verify multiple repositories simultaneously
- Example Use Case: Verify all repositories from a specific user (e.g., "microsoft/")
- Enhance verification system: Not only verify Repos, also verify users/developers.

10.4.2 Frontend Improvements

Responsive Design and Layout

- Implement a mobile-first responsive design approach
- Ensure seamless user experience across different device types:
  - Smartphones
  - Tablets
  - Desktop computers
- Develop an adaptive layout that dynamically adjusts to screen sizes

Wiki Page

- Implement a wiki page to explain website features
- Show Healthy Repository Definition in wiki
- Foundation Registration Interface: Better descriptions what multiplier options are

Dashboard Enhancements

- Expand repo card display to show more repositories
- Implement toggle functionality between:
  - Card view (detailed visualization)
  - List view (compact information display)

Foundation View - Repository Selection Experience

- Current Problem: Multiple repository selection leads to:
  - Vertical page expansion
  - Visual clutter
  - Reduced user experience
- Recommended Solution: Develop a more compact, manageable selection interface

Navigation and Scrolling Optimization

- Fix header and navigation bar to remain static
- Implement scrolling functionality only for the main content area
- Ensure consistent navigation experience across all pages

10.4.3 Payment and Sponsorship Enhancements

Credit Card Management

- Limit payment method to a single credit card and crypto currencies
- Redesign payment interface to:
  - Separate donation button visually from other payment elements
  - Improve clarity and user experience of payment process

Dynamic Cryptocurrency Sponsorship Limit

- Implement dynamic daily multiplier sponsoring limit for cryptocurrency
- Develop real-time price fetching mechanism:
  - Integrate with cryptocurrency price API
  - Use current market price as reference for sponsorship limits
  - Account for cryptocurrency price volatility
- Automatically adjust daily multiplier sponsoring limit based on current exchange rates

10.4.4 Testing Improvements

Admin Test Page Enhancements

- Extend existing test page functionality
- Add capability to generate fake data for:
  - Foundation creations (with daily multiplier sponsoring limit)
  - Foundation contribution to repo or single contributor
- Improve test data generation to cover edge cases and complex scenarios

10.4.5 Multiplier Calculation

The code that calculates fee multipliers has a TODO comment about making the data tracking better. Currently, when calculations are made for a data record, it just modifies the existing record. This means we don't have a history of it. If we fix this, we could:

- Keep track of old records instead of deleting them
- Count how many records we've created over time
- See how records have changed from start to finish

This change would make it easier to check what's happened in the system. As the code is fairly straight forward, the fix to that is not too hard.

10.4.6 API Improvements

At date of term project hand-in the API endpoint `"GET /repos/trusted"` is used to call the backend, to get all verified repositories. This could lead to an extremely large API call, when the platform gets bigger and thousands or millions of repos are verified by FFS.
As a solution, we propose to add repo verification as a property of the Repo struct. In this case the verified information is transmitted within the Repo object itself, which is fetched anyways.

The same issue is faced with the API endpoint `"GET /repos/id/multiplierCount"`, this can be improved in the same way.

10.5 Lessons Learned

The project reinforced several crucial project management principles. Comprehensive planning proved essential for successful delivery, providing a clear roadmap while maintaining flexibility for adjustments. Regular stakeholder meetings emerged as a vital component, enabling continuous alignment on implementation decisions and project direction through consistent updates and feedback cycles.

Team dynamics played a pivotal role in project success. REgular communication and mutual support helped overcome technical challenges, particularly while learning new technologies. We learned that early task prioritization is crucial - our initial delays in establishing clear priorities highlighted how this impacts project flow. This experience emphasized the importance of defining and tackling critical components first to maintain steady progress throughout the development cycle.

Analysis of our sprint time tracking, as illustrated in Figure 10.5, reveals a significant disparity between the first and second halves of the project. The second half shows substantially higher average time investment per sprint, which can be attributed to two key factors: insufficient initial preparation and unclear strategic direction. This highlights a critical lesson learned: securing all necessary project materials early and establishing a clear project direction at the outset is essential for maintaining consistent resource utilization. Early investment in project planning and strategic alignment could have prevented the increased time demands in later sprints and led to more efficient resource allocation throughout the project lifecycle.

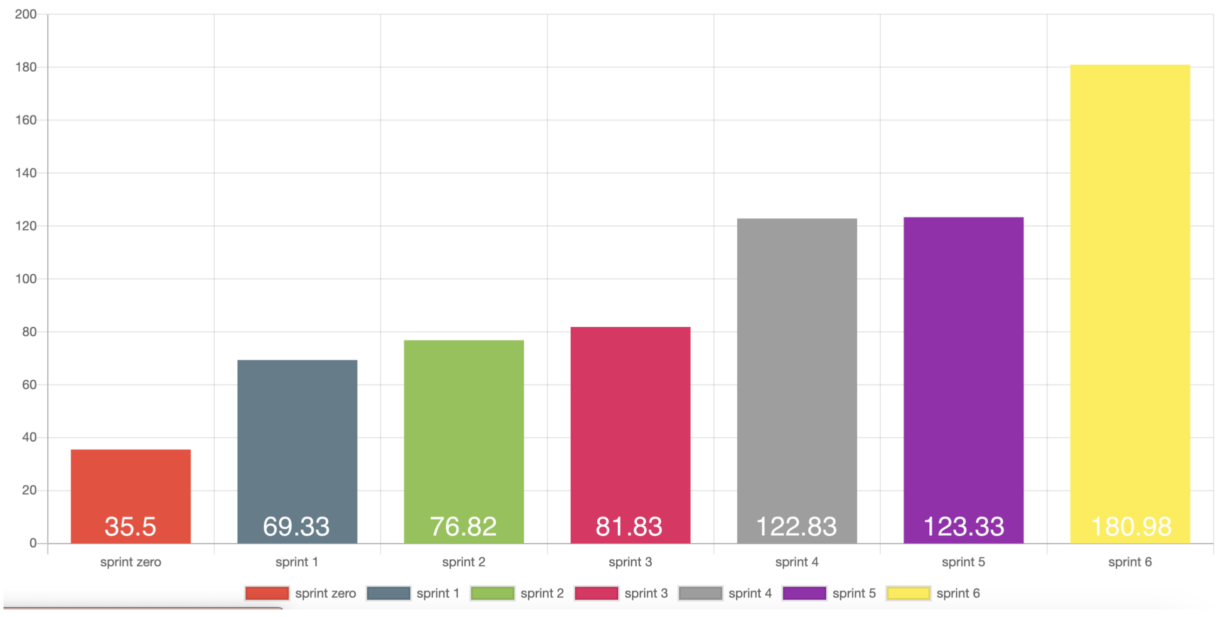


Figure 10.5: Sprint Overview

Part V

Project Documentation

Project name: flatfeestack

Team Members

1. Yannick Staedeli (yannick.staedeli@ost.ch)
2. Mino Petrizzo (mino.petrizzo@ost.ch)
3. Roman Cvijanovic (roman.cvijanovic@ost.ch)

Availabilities

Time table meeting

Time slot	Mon	Tue	Wed	Thu	Fri
08h00-09h00	XR	XO	-	XR	-
09h00-10h00	XR	XO	-	XR	-
10h00-11h00	XR	XO	-	XR	-
11h00-12h00	XR	XO	-	XR	-
12h00-13h00	XR	XO	-	XR	-
13h00-14h00	XR	XO	-	XR	-
14h00-15h00	XR	-	-	-	-
15h00-16h00	XR	-	-	-	-
16h00-17h00	XR	-	-	-	-
17h00-18h00	XR	-	-	-	-
18h00-19h00	XR	-	-	-	-

Legend

XO	Slot available online
XR	Slot available in Rapperswil

Chapter 11

Project Plan

11.1 Role distribution and collaboration framework

We have decided to work with the SCRUM method. This allows us to remain flexible and develop a good product as quickly as possible. Accordingly, we have defined the following roles and divided them among ourselves:

11.1.1 Role Distribution

Role	**Members**	**Explanation**
Product Owner	Thomas Bocek	As the product owner, Thomas Bocek will steer us in the correct direction with his desired implementation.
Project Leader	Thomas Bocek	As the project Leader, Thomas Bocek will be the tie breaker if the team finds no common ground.
Stakeholder	Thomas Bocek	As the main stakeholder and owner of flatfeestack, Thomas Bocek will want the main goal of multiplier activation be fulfilled primarily.
Scrum Master	Roman Cvijanovic	As the Scrum Master, Roman Cvijanovic will lead the Sprint Meetings, keep an overview of feature progress and keep the team informed.
Developer	Yannick Staedeli Mino Petrizzo Roman Cvijanovic	As developers, everyone is invested to develop a product that satisfies current standards and the Stakeholder's requirements.

Table 11.1: Role distribution

11.1.2 Roles scope

It is important to keep track of the scope each role has to cover, so there will not be the problem of certain work being done double, and other work not being done at all. We have to clearly define, who takes care of what.

Product Owner

The scope of a Product Owner (PO) typically involves various responsibilities and activities throughout the product development life-cycle. Here are some key aspects of the scope of a Product Owner:

1. **Defining product vision:** The Product Owner is responsible for defining and communicating the overall vision for the product.
2. **Progress Checkups:** The product owner will check on our progress and ensure we are steering in the right direction.

Project Leader

1. **Decision Maker:** In case if divergent opinions the Project Leader has the ultimate vote.

Stakeholder

The stakeholder is the main beneficiary of the solution of this project.

1. **Conveying wishes:** As the main beneficiary the stakeholder states the goals to be achieved mostly in an abstract manner.

Scrum Master

The scope of a Scrum Master involves various responsibilities and activities aimed at facilitating the Scrum framework's successful implementation and ensuring the team's effectiveness. Here are key aspects of the Scrum Master's scope:

1. **Facilitating Scrum Events:** The Scrum Master facilitates various Scrum events, including Sprint Planning, Daily Stand-ups, Sprint Reviews, and Sprint Retrospectives. They ensure that these events are conducted effectively, time-boxed, and focused on achieving their objectives.
2. **Removing Impediments:** The Scrum Master identifies and removes impediments that hinder the team's progress. This involves addressing issues such as organizational barriers, resource constraints, and conflicts within or outside the team.
3. **Coaching and Mentoring:** The Scrum Master coaches the Scrum Team and Product Owner on Scrum principles, practices, and values. They help individuals and teams understand and adopt Scrum roles, artifacts, and ceremonies.
4. **Journal:** The Scrum Master journals sprints and meetings in an open manner and shares everything transparently.

Developer

In this project, the Developer role encompasses multiple technical responsibilities. Instead of creating separate roles for closely related tasks, we've consolidated these responsibilities under the Developer role. This approach allows for flexible task distribution among team members throughout the project lifecycle.

1. **Code Development** Primary responsibility involves implementing features, writing maintainable code, creating comprehensive unit tests, and deploying completed functionality to the production environment.
2. **Testing** Due to our streamlined team structure, developers perform testing duties during Pull Request reviews at sprint boundaries. This includes integration testing, regression testing, and verifying acceptance criteria.
3. **Quality Assurance** Developers are responsible for code quality enforcement through peer reviews, static code analysis, and adherence to coding standards. This includes monitoring code coverage, identifying technical debt, and ensuring documentation completeness.

11.2 Project Planning Tracking

Managing task allocation and progress tracking in complex projects like FlatFeeStack requires robust project management infrastructure. Our team has implemented JIRA as our primary project management tool. We selected JIRA based on its comprehensive feature set and the team's existing proficiency from prior development projects.
JIRA provides essential capabilities for our development workflow:

- Agile boards for sprint planning and execution
- Task tracking with customizable workflows
- Integration with version control systems
- Story point estimation and velocity tracking
- Automated reporting and metrics collection
- Configurable issue types and fields

The team's familiarity with JIRA's interface and functionality enables immediate productive use without additional training overhead. This standardization on JIRA streamlines our project management processes and facilitates effective sprint management.

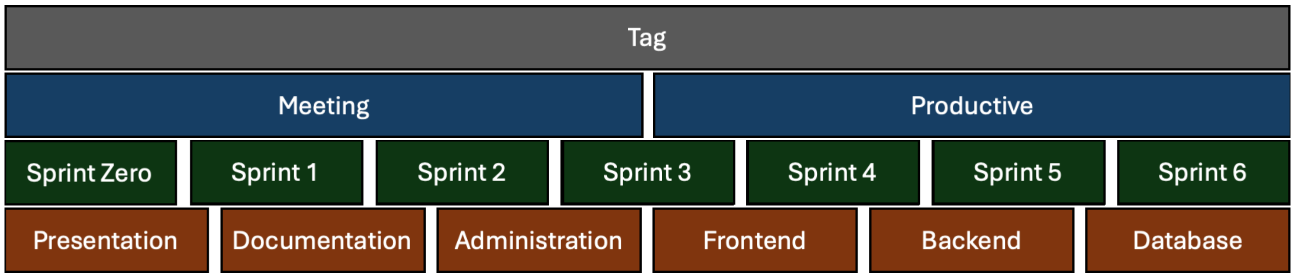
Chapter 12

Time Tracking

To provide an accurate time tracking, we use an additional app in Jira called "Timetracker". Every task for our project is documented within our issue management system Jira. This systematic approach logically extends to time tracking, where we record the duration spent on each issue. We must also account for the time spent in meetings, as it constitutes a significant portion of our project efforts. To facilitate this, we create issues in the Backlog for different meeting types, titled accordingly. Every participant is then required to log their time against this issue.

How to Tag the Time Logs

Ultimately, we aim to provide a precise overview of the hours invested in our project. This schema clearly illustrates how we tag our time logs. **Each time log must contain one tag from each row.** If we follow that principle, we can generate a chart based on a row of tags. Each row of tags in the scheme represents the total time spent on the project.



12.1 Reports

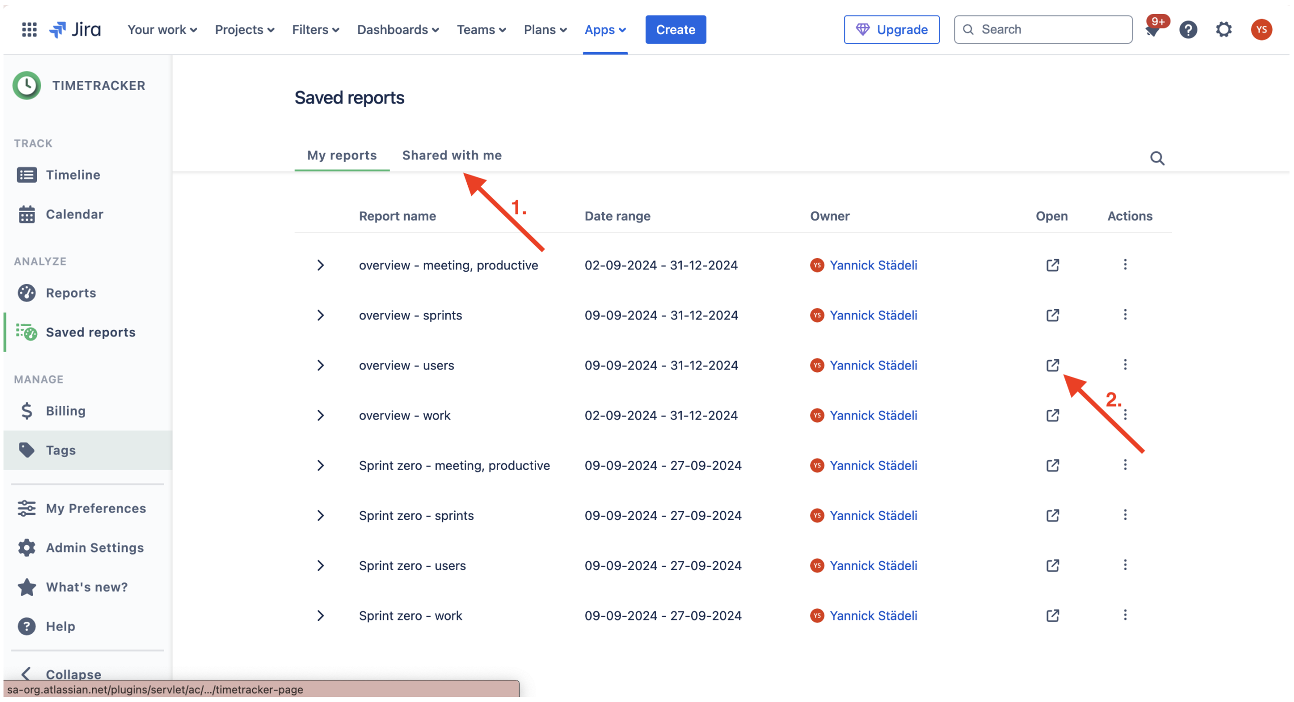
To easily generate reports, we created predefined filters with the tags described. The reports can also be viewed in the time tracking application at the following link. The released reports always refer to the total project duration, allowing you to see the current hours at any time. This visualization of our time tracking makes it easy to verify the progress.

Generate Time Reports per Person per Task

To gain a detailed understanding of how many hours each person has invested in specific tasks, refer to our time-tracking application:

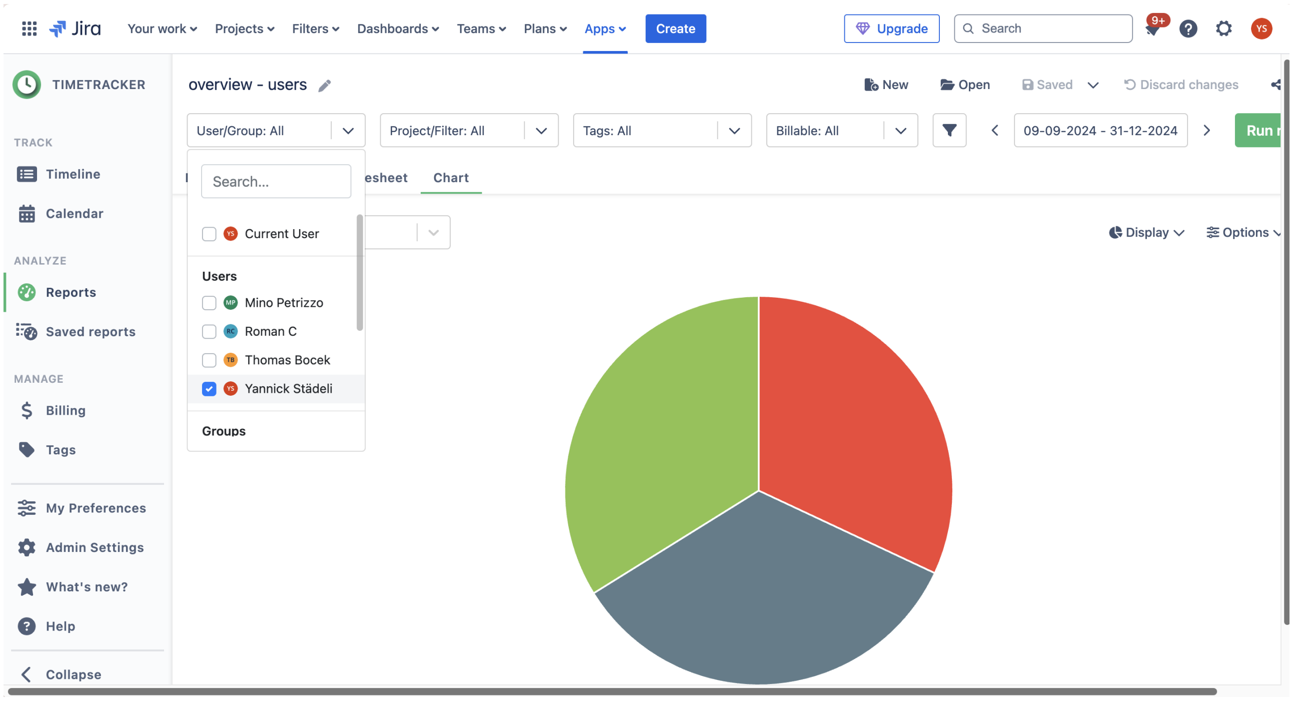
Use the link above, then follow the screenshots...

Navigate to the *Shared with me* tab, there you will see the report templates we created and shared with all project members.



Report name	Date range	Owner	Open	Actions
overview - meeting, productive	02-09-2024 - 31-12-2024	Yannick Städeli	↗	⋮
overview - sprints	09-09-2024 - 31-12-2024	Yannick Städeli	↗	⋮
overview - users	09-09-2024 - 31-12-2024	Yannick Städeli	↗	⋮
overview - work	02-09-2024 - 31-12-2024	Yannick Städeli	↗	⋮
Sprint zero - meeting, productive	09-09-2024 - 27-09-2024	Yannick Städeli	↗	⋮
Sprint zero - sprints	09-09-2024 - 27-09-2024	Yannick Städeli	↗	⋮
Sprint zero - users	09-09-2024 - 27-09-2024	Yannick Städeli	↗	⋮
Sprint zero - work	09-09-2024 - 27-09-2024	Yannick Städeli	↗	⋮

Figure 12.1: Generate Time Reports - step 1



overview - users

User/Group: All

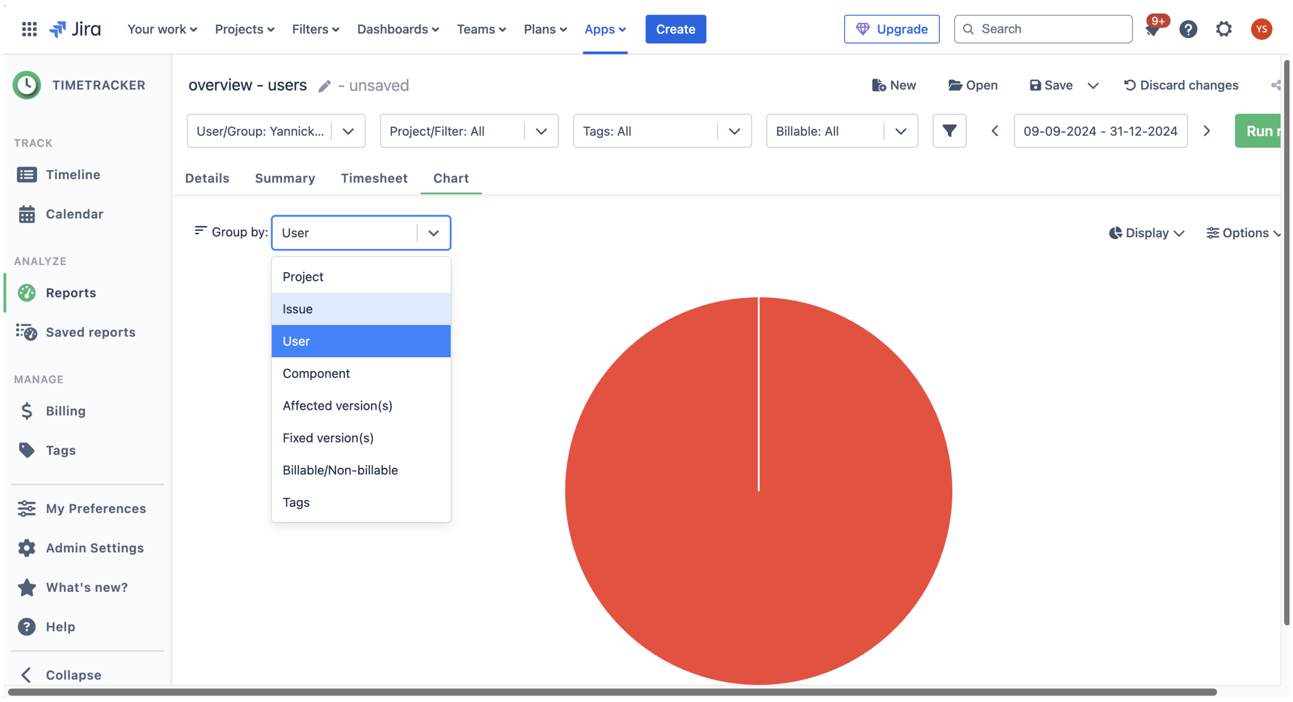
Project/Filter: All

Tags: All

Billable: All

09-09-2024 - 31-12-2024

Figure 12.2: Generate Time Reports - step 2



overview - users - unsaved

User/Group: Yannick...

Project/Filter: All

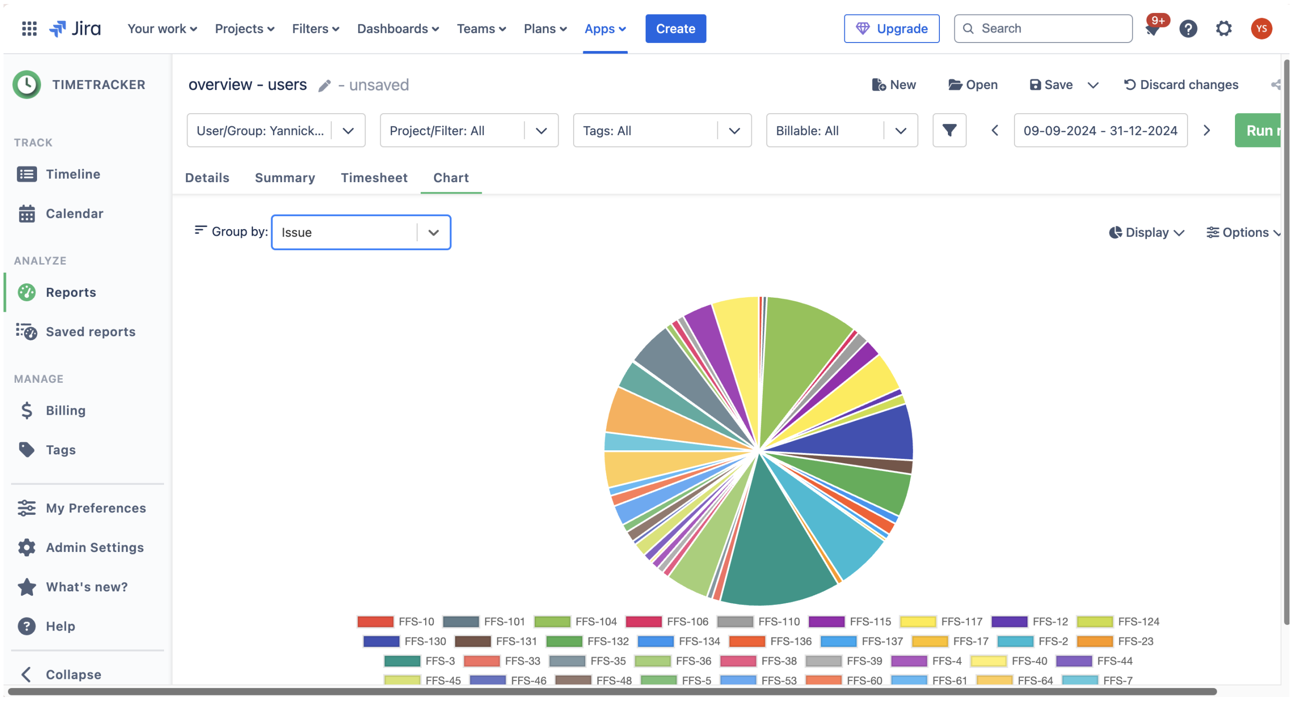
Tags: All

Billable: All

09-09-2024 - 31-12-2024

Group by: User

Figure 12.3: Generate Time Reports - step 3



overview - users - unsaved

User/Group: Yannick... | Project/Filter: All | Tags: All | Billable: All | 09-09-2024 - 31-12-2024

Group by: Issue

Figure 12.4: Generate Time Reports - step 4

The final reports at project end with all detailed visuals can be found in the appendix: Time Tracking Reports

Chapter 13

Technical Debts and Open Debts

Throughout the whole project it's highly likely we'll have unsatisfactory solutions or situations. These elements are called technical debts and they will be listed in a simple table. These technical debts are always referenced to JIRA items.

The open debts can simply be regarded as some general tasks that are organizational and not technical debts. Usually these tasks should have already been finished but because of time constraints they aren't done on time.

13.1 Technical Debts

13.2 Open Tasks

Bibliography

[Bue24] Ivan Buetler. Personal interview with ivan buetler, 2024. Conducted on October 30, 2024.

[Cla24] Claude. Claude: Conversational ai model and image generator. https://claude.ai/, 2024. Last accessed: 2024-12-19.

[Dev24] Practical DevSecOps. Threat modeling vs risk assessment: Understanding the difference. https://www.practical-devsecops.com/threat-modeling-vs-risk-assessment/, 2024. Last accessed: 2024-11-18.

[Fig24] Inc. Figma. Figma: Collaborative interface design tool. https://www.figma.com, 2024. Last accessed: 2024-11-25.

[Fou24] OWASP Foundation. Threat modeling. https://owasp.org/www-community/Threat_Modeling, 2024. Last accessed: 2024-11-18.

[ISA22] ISACA. An integrated approach to security audits. https://www.isaca.org/resources/news-and-trends/industry-news/2022/an-integrated-approach-to-security-audits, 2022. Last accessed: 2024-11-18.

[LdA24] JGraph Ltd and draw.io AG. diagrams.net (formerly draw.io): Online drawing tool. https://app.diagrams.net/, 2024. Last accessed: 2024-12-19.

[Mar09] Robert C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. Robert C. Martin Series. Prentice Hall, 2009.

[MHF$^+$24] Livio Mauchle, Simon Hefti, Martyn Foreman, Yannick Städeli, and Luca Köppel. Swisscardgames: Final documentation. Technical report, School of Computer Science, OST Eastern Switzerland University of Applied Sciences, 2024.

[moq24] moqups. Moqups von evercoder software srl. https://moqups.com/de/, 2024. Last accessed: 2024-11-30.

[Ope24] OpenAI. Chatgpt: Conversational ai model and image generator. https://openai.com/chatgpt, 2024. Last accessed: 2024-12-19.

[Sve] Svelte. Svelte tutorial. https://svelte.dev/tutorial/svelte/welcome-to-svelte. Last accessed: 2024-09-23.

[W3Sa] W3Schools. Css dropdowns tutorial. https://www.w3schools.com/css/css_dropdowns.asp. Last accessed: 2024-11-16.

[W3Sb] W3Schools. Css toggle switch tutorial. https://www.w3schools.com/howto/howto_css_switch.asp. Last accessed: 2024-11-02.

List of Figures

List of Tables

List of Algorithms

Part VI

Appendix

Chapter 14

Appendix

14.1 Test Reports

14.1.1 Acceptance Test Reports

QA Team

Participant’s Name	Yannick Städeli, Mino Petrizzo, Roman Cvijanovic
Date	December 12, 2024
Location	Remote
Tasks	FR1: Select Git Project → Participant Feedback: - FR2: View Multiplier Sponsoring History → Participant Feedback: Date is missing in the payment table. Test is missing to create fake multiplier sponsoring. FR3: Toggle Multiplier → Participant Feedback: It Works in the frontend, backend test are implemented. But again, no way to test it in the frontend with fake data. FR4: Apply Multiplier → Participant Feedback: Not exactly correct implemented. Now, when multiplier money to pay exceeds the limit, nothing is payed. But, it should pay a fraction based on the rate between is/should. FR5: View Multiplier Status → Participant Feedback: - FR6: Daily Multiplier Sponsoring Limit → Participant Feedback: - NFR1: Multiplier Accuracy → Participant Feedback: - NFR2: Performance under Concurrent Use → Participant Feedback: Was not actively tested, we think it is not very important right now. NFR3: Dynamic Multiplier Calculation → Participant Feedback: - NFR4: Secure Multiplier Access → Participant Feedback: -
Overall Feedback	Most of the features work, some stuff mentioned in the FRs are not implemented and we have to take action.
Status	Done

Table 14.1: Acceptance Test Protocol - QA Team

Dr. Thomas Bocek

Participant's Name, Job / Degree	Dr. Prof. Thomas Bocek - Computer Science
Browser / Device	Starlite
Date	December 12, 2024
Location	Remote
Tasks	FR1: Select Git Project → Participant Feedback: Everything works. Side observation: Toggle off and on then the multipliers active on repos are active like before. FR2: View Multiplier Sponsoring History → Participant Feedback: remove text above input field for daily limit: only input field and then show current limit in field with bind value. Feedback to payment page: need a better solution for adding credit card and pay the donation. FR3: Toggle Multiplier → Participant Feedback: It's ok. FR4: Apply Multiplier → Participant Feedback: It's ok. FR5: View Multiplier Status → Participant Feedback: It's ok. FR6: Daily Multiplier Sponsoring Limit → Participant Feedback: The Limit only provides an input field for dollar, what happens when a someone pay in crypto currency? Our solution is, always set limit in dollar, show what amount in crypto it is with "live" (updated daily) exchange rate. NFR1: Multiplier Accuracy → Participant Feedback: Is's ok. NFR2: Performance under Concurrent Use → Participant Feedback: Not very important, can be neglected. NFR3: Dynamic Multiplier Calculation → Participant Feedback: update NFR3 description –> mention the trust and untrust mechanism NFR4: Secure Multiplier Access → Participant Feedback: It's ok.
Overall Feedback	-
Status	Done

Table 14.2: Acceptance Test Protocol - Dr. Thomas Bocek

14.1.2 Usability Test Reports

Participant's Name, Job / Degree	
Browser / Device	TBD
Date	December 5, 2024
Location	OST Rapperswil
Facilitator	Yannick Städeli
Tasks	Test Case Admin 1 → Observations: TBD → Participant Feedback: TBD Test Case Admin 2 → Observations: TBD → Participant Feedback: TBD Test Case Admin 3 → Observations: TBD → Participant Feedback: TBD Test Case Admin 4 → Observations: TBD → Participant Feedback: TBD Test Case Admin 5 → Observations: TBD → Participant Feedback: TBD Test Case Admin 6 → Observations: TBD → Participant Feedback: TBD Test Case Admin 7 → Observations: TBD → Participant Feedback: TBD Test Case Admin 8 → Observations: TBD → Participant Feedback: TBD
Expected Results	Defined in Test Cases
Overall Feedback	TBD
Status	TBD

User / Foundation - Participant 1

Participant’s Name, Job / Degree	Michele Petrizzo, Technical Designer, higher technical college
Browser / Device	Google Chrome / MacBook Pro 16" M3 Pro
Date	December 10, 2024
Location	Remote
Facilitator	Yannick Städeli
Tasks	Test Case User 1 → Observations: He quickly found out, to activate the multiplier, he has to go to the settings. By reading the information about the multiplier on the landing page he already knew without the helper how the sign looks and saw it in the search. → Participant Feedback: The daily limit should also be set by entering an amount and focus out of the input, instead of enter or button. Else everything was clear and understandable he said. Test Case User 2 → Observations: He had to figure it out, that in the search he is able to only set the star and multiplier, and on top he is able to remove them, but he understood it after 2-3 seconds. → Participant Feedback: He said that it makes sense because when he searches for another repo after he activated the multiplier on a repo, he can easily remove it on top again. Test Case User 3 → Observations: He saw it straight away and understood that the top cards directly show which repos are sponsored and which also have a multiplier on them. → Participant Feedback: For a small number of repos this is a good thing, but if there are more he sees this critically because the site then gets bigger and bigger. Test Case User 4 → Observations: He saw the icon for the healthy repos and because I had healthy and unhealthy repos, he understood which were the healthy ones. → Participant Feedback: He understood which repos are healthy but not why they are healthy, for him it was something like a help to know that his money is most likely good spent.
Expected Results	Defined in Test Cases
Actual Results	He was able to fulfill all 4 tests without much help and only addressed minor deficiencies for him. However, I can tick each test off as passed.
Overall Feedback	He gave some good inputs, and if this happens also to other participants its something to consider
Status	Done

User / Foundation - Participant 2

Participant's Name, Job / Degree	Florian Asaj, Federal Vocational Baccalaureate
Browser / Device	Firefox / Custom PC
Date	December 10, 2024
Location	Remote
Facilitator	Mino Petrizzo
Tasks	Test Case User 1 → Observations: Did not find it on first try but after looking trough the web application he saw it under the settings and understood fast that the changes are in the search. → Participant Feedback: He wanted a better guidance to the multiplier enable switch. Test Case User 2 → Observations: Saw where the icon is clickable and where not. → Participant Feedback: He understood it right away where to click to enable and disable the multiplier / sponsoring. Test Case User 3 → Observations: The test was maybe to simple because he already saw it inside the search but thought that there has to be another page for it. → Participant Feedback: The sponsored and multiplier repos should not be inside the search, because it has nothing to do with a search. Test Case User 4 → Observations: Because of social media he already knew that the blue icon on the right side is the one to indicate healthy and unhealthy (verified and unverified) repos. → Participant Feedback: That is understandable and clear to him.
Expected Results	Defined in Test Cases
Actual Results	Good, I can say that he has passed every test.
Overall Feedback	Clear, simple design and a very minimalist focus on the essentials.
Status	Done

Table 14.4: User / Foundation Test Protocol - Participant 2

User / Foundation - Participant 3

Participant’s Name, Job / Degree	Sarah Petrizzo, Commercial Apprenticeship - Student at ZHAW
Browser / Device	Google Chrome / MacBook Pro 16" M3 Pro
Date	December 10, 2024
Location	Remote
Facilitator	Mino Petrizzo
Tasks	Test Case User 1 → Observations: First did not go in the settings and needed a tip to find it. → Participant Feedback: At the beginning it was not quite clear that the switch for foundations was in the settings, but in retrospect it makes sense. Test Case User 2 → Observations: She searched a project and thanks to the helper in the settings she understood the multiplier icon and needed a moment to understand where to add and remove the multiplier. → Participant Feedback: She said that, she does not understand why its not possible to remove the multiplier in the search too. Test Case User 3 → Observations: After I told the test goal she looked at the cards on top and saw it right away → Participant Feedback: The overview on top is a good thing, so that every sponsored and multiplied repo is visible right away. Test Case User 4 → Observations: Saw that on every search result there was a blue icon and understood thanks to social media that this is an verified sign for healthy repos. → Participant Feedback: This is also a good solution and easy to understand I like it, she said.
Expected Results	Defined in Test Cases
Actual Results	Mostly positive all test cases can be marked as passed and she had just some suggestions to make it more understandable.
Overall Feedback	Its mostly simple to use and does not take to much time.
Status	Done

Table 14.5: User / Foundation Test Protocol - Participant 3

Admin - Participant 1

Participant's Name, Job / Degree	Livio Mauchle, Bsc Computer Science Student
Browser / Device	Microsoft Edge / MacBook Pro 16" M3 Pro
Date	December 09, 2024
Location	Remote
Facilitator	Yannick Städeli
Tasks 1 - 4	Test Case Admin 1 → Observations: Livio was a little confused when the button to mark repo as healthy was deactivated at first. He then realized by himself to first analyze it, then mark it as healthy. → Participant Feedback: He said it is good solution to have the ability to undo it when removing a repo from the list. Test Case Admin 2 → Observations: I observed he tested one edge case with the health buttons; change the page during an undo progress. It worked fine. → Participant Feedback: Good to cover such edge cases. Test Case Admin 3 → Observations: After the first two test cases he was already a little familiar with the platform. It was clear to him, to change to the "Search" tab and search for the repo as a user. → Participant Feedback: That verify star is a very common way to signalize that something is verified or in our case "healthy" Test Case Admin 4 → Observations: He already clicked through the repo assessment overlay when he was searching to untrust a repo, so he exactly knew where to find the score. → Participant Feedback: -

Participant’s Name, Job / Degree	Livio Mauchle, Bsc Computer Science - Student at OST
Tasks 5 - 8	Test Case Admin 5 → Observations: For Livio it was not clear what score is best, high number or low number. Therefore i had to help him with this a little, and tell him that 10 is the maximum score. → Participant Feedback: Good to sort the repos, makes it easy to remove unmaintained repos. Test Case Admin 6 → Observations: During one of the previous testcases, he already opened the "Threshold Settings", so when i told him to find the used Thresholds, he associated this part of the website. What i meant, was the Assessment Overlay, where you can see the thresholds graphically. → Participant Feedback: It was not so good described from my side, which is why he ended up on the wrong page. But after clearing thins up, he thought that everything makes sense how it is. Test Case Admin 7 → Observations: He instantly navigated to the repo assessment of one repo and told me the partial health values for each metric. → Participant Feedback: nice to have an overview and a detailed graph. Test Case Admin 8 → Observations: After the first seven Testcases Livio was already a very good FFS admin and set the new threshold in a few seconds → Participant Feedback: Very good idea to manually set the thresholds.
Expected Results	Defined in Test Cases
Overall Feedback	The first few moments on the Healthy Repos page, he immediately searched for a repo. He then got a little confused about the healthy repos at the top and the one repos found through the search tab. So maybe there are a little to much information when using the platform the first time. But he was familiar with it very fast.
Status	Done

Table 14.6: Admin Test Protocol - Participant 1

Admin - Participant 2

Participant's Name, Job / Degree	Rico Staedeli, Msc Computer Science - Student at HSG
Browser / Device	Safari / MacBook Pro 16", M3 Max
Date	December 10, 2024
Location	Remote
Facilitator	Yannick Städeli
Tasks	For this participant i chose another approach to make the tests. Before starting with the actual testcases i let him "play" with the platform to get a little familiar with it. After telling him his role, he found every feature by himself. Test Case Admin 1 → Observations: No Problem to figure it out by himself. → Participant Feedback: Trust Icon makes sense, it is very easy to understand what it does. Test Case Admin 2 → Observations: not actively tested with him. → Participant Feedback: - Test Case Admin 3 → Observations: He had concerns that when he marks a repo with a bad score as healthy, that it might not be stared from users. He then realized, that the Health Value is not visible for the users. → Participant Feedback: It is good to not show, the Health Value to the users. Because when he trust a repo of a trustworthy friend with a bad score, it has no impact users decisions. Test Case Admin 4 → Observations: Easy to find, for him the page partition was clear. → Participant Feedback: -

Participant’s Name, Job / Degree	Rico Staedeli, Msc Computer Science - Student at HSG
Tasks	Test Case Admin 5 → Observations: Rico found this feature already during the first phase of testing. For him it was clear that low score means bad, and high score means good. → Participant Feedback: He thinks the overview does not work so well of him. He said he would never make horizontal scroll bar. But with the search and sort functions it is still easy to use. Test Case Admin 6 → Observations: He was searching for it in threshold settings. → Participant Feedback: It would be great to have a more detailed what the metrics are in the threshold settings as well as in the repo assessment. Test Case Admin 7 → Observations: He knew right away where to click, as he already viewed this page in the previous test cases. → Participant Feedback: At first sight it was not logic for him to click on the Health Value. It does not look like a button for him. Hes recommendation is to add a separate button called "Details" to show the repo assessment. Test Case Admin 8 → Observations: Easy to find for him → Participant Feedback: At first Rico said he wishes to have different thresholds for different sizes of repositories. But when i told him the users never actually see the Health Value it ok for him.
Expected Results	Defined in Test Cases
Actual Results	It was not so clear to him what the meaning of each metric name was.
Overall Feedback	It is a very nice project, he thinks after using this admin page once it makes sense how to use it. He does not miss any feature. To make the healthy repos dashboard more convenient to use, he would move the "Add new Healthy Repositories" to the top of the page and the healthy dashboard to the bottom. It would allow to see more of the healthy repos at once, which is important for him.
Status	Done

Table 14.7: Admin Test Protocol - Participant 2

Participant's Name, Job / Degree	Norbert Knobel, SysAdmin, EFZ
Browser / Device	Linux Mint / Firefox
Date	December 11, 2024
Location	Remote
Facilitator	Roman Cvijanovic
Tasks	Test Case Admin 1 → Observations: Executed task successfully. → Participant Feedback: Not intuitive as to why trusting is not possible immediately. First analyse repo, then trust it seems weird. Five seconds seemed a bit long for participant. Test Case Admin 2 → Observations: Executed task successfully. → Participant Feedback: Good instant feedback. Test Case Admin 3 → Observations: An error trying to reselect a unhealthy repo health again occurred. Error couldn't not be replicated imitating the steps taking by user, thus task successful. → Participant Feedback: Confusing that there are two search views. Test Case Admin 4 → Observations: Executed task successfully. → Participant Feedback: Nice overview of the metrics and how the points are distributed. Test Case Admin 5 → Observations: Executed task successfully. → Participant Feedback: Repo Card window small and side scrolling is not intuitive. Spread of points is bad, as their are either 0 or 4. Test Case Admin 6 → Observations: Executed task successfully. → Participant Feedback: . Test Case Admin 7 → Observations: Executed task successfully. → Participant Feedback: TBD Test Case Admin 8 → Observations: Executed task successfully. → Participant Feedback: TBD
Expected Results	Defined in Test Cases
Actual Results	TBD
Overall Feedback	TBD
Status	TBD

Table 14.8: Result for Usability Test Protocol with FFS Admin 1

Participant’s Name, Job / Degree	Mario Cvijanovic, Freelancer, EFZ
Browser / Device	Brave / Windows 11
Date	December 11, 2024
Location	Remote
Facilitator	Roman Cvijanovic
Tasks	Test Case Admin 1 → Observations: Executed task successfully but with issues. Two search bars confused user. → Participant Feedback: Button and name are too close to each other. Test Case Admin 2 → Observations: Executed task successfully. → Participant Feedback: No comment. Test Case Admin 3 → Observations: Executed task successfully. → Participant Feedback: No comment. Test Case Admin 4 → Observations: Executed task successfully with help. User didn’t know that Repo Health Value is interactive. → Participant Feedback: Maybe add some kind of help page. Test Case Admin 5 → Observations: Executed task successfully. Polling confused user, but page refresh cleared confusion. → Participant Feedback: No comments. Test Case Admin 6 → Observations: Executed task successfully. User needed suggestion to click on either the repo assessment button or on the gear button. → Participant Feedback: No comments. Test Case Admin 7 → Observations: Executed task successfully. → Participant Feedback: No comments. Test Case Admin 8 → Observations: Executed task successfully. User was able to insert a floating point number - bug. → Participant Feedback: No comments.
Expected Results	Defined in Test Cases
Actual Result	Mostly successfully.
Overall Feedback	In general good, but some improvements necessary.
Status	Done

Table 14.9: Result for Usability Test Protocol with FFS Admin 2

14.2 Original Assignment

The original assignment is available at the AVT website of OST.

14.2.1 Beschreibung der Arbeit

Intro

Viele Projekte und Produkte enthalten Open-Source-Software, und es ist entscheidend, dass diese Projekte ihre Software kontinuierlich verbessern, weiterentwickeln, pflegen und Fehler beheben. Nicht gewartete Software-Bibliotheken oder Frameworks stellen ein Sicherheitsrisiko dar. Spenden bieten einen Anreiz, diese Bibliotheken und Frameworks zu pflegen und Fehler zu beheben.

Spendendienste wie Open Collective, Patreon oder GitHub Sponsoren ermoeglichen es, Projekte oder Personen direkt zu unterstuetzen, beispielsweise Projekt X mit $120 pro Jahr oder Person Y mit $120 pro Jahr. Der Entscheidungsprozess, welche Projekte oder Personen unterstuetzt werden sollen und in welchem Umfang, ist jedoch zeitaufwaendig und oft intransparent (warum wird Projekt X unterstuetzt und nicht Projekt Y?).

Mit FlatFeeStack wird das Spenden einfacher und transparenter. Fuer $120 pro Jahr kann jeder Entwickler ein beliebiges Open-Source-Projekt unterstuetzen – die Spende wird gleichmaessig auf die Projekte aufgeteilt. Fuer Unternehmen ist dies einfacher zu budgetieren, da es sich um eine Pauschalgebuehr pro Entwickler handelt. Die Verteilung der Mittel an die Open-Source-Contributors richtet sich danach, wie viel Code der Entwickler geaendert hat.

Projektseite: https://github.com/flatfeestack

Ziele

In dieser Arbeit soll ein Multiplikator innerhalb von Flatfeestack implementiert werden, der es ermoeglicht, diverse Projekte mit einem Multiplikator zu unterstuetzen. Z.B. Eine Person Y moechte Projekt X mit einem Multiplikator 2 unterstuetzen, wenn Projekt X 123$ Spenden bekommt (von beliebigen Spendern), dann bekommt Projekt X mit dem Multiplikator insgesamt 246$, wobei die 123$ von Person Y kommt. Ein grosses Problem besteht jedoch darin, dass Multiplikatoren ausgenutzt werden koennen. Beispielsweise koennte ein Entwickler Y einen Multiplikator von 3 auf Projekt X anwenden. Wenn Y nun 100 CHF in das Projekt investiert, erhaelt er 300 CHF zurueck, von denen er 100 CHF selbst eingesetzt hat, sodass er einen Gewinn von 200 CHF erzielt. Dies stellt ein erhebliches Risiko dar, da es die Integritaet des Finanzierungssystems untergraben kann.

Das Ziel dieser Arbeit ist es, einen automatisierten Mechanismus zu entwerfen und zu implementieren, der verhindert, dass Multiplikatoren missbraeuchlich verwendet werden. Gleichzeitig soll das Multiplikator-Feature beibehalten werden, um die Unterstuetzung von Open-Source-Projekten weiterhin attraktiv zu gestalten. Der Mechanismus soll sicherstellen, dass Multiplikatoren fair und transparent eingesetzt werden, ohne dass Einzelpersonen oder Projekte durch unethisches Verhalten profitieren koennen.

14.2.2 Anforderungen an Studierende

Anforderungen / Techstack

Neben der Implementierung in Golang im Backend muss zunaechst ein Mechanismus entworfen werden. Dieser Mechanismus soll sicherstellen, dass die Multiplikatoren gerecht und ohne Missbrauchsmoeglichkeiten angewendet werden koennen. Dabei werden verschiedene Algorithmen und Sicherheitsmassnahmen entwickelt, um zu ueberpruefen, ob die eingesetzten Gelder legitim sind und keine betruegerischen Aktivitaeten stattfinden. Der Entwurf dieses Mechanismus ist entscheidend, um eine robuste und zuverlaessige Loesung zu gewaehrleisten, die das Vertrauen der Nutzer in das Multiplikator-Feature staerkt und gleichzeitig die Integritaet der finanziellen Unterstuetzung fuer Open-Source-Projekte wahrt. Dieser Mechanismus soll in die bestehende Flatfeestack Applikation integriert werden. Flatfeestack verwendet im Frontend Svelte.

14.3 Time Tracking Reports

Information about the Visuals

- All time information in the visuals are in hours.
- The colors of the bars may change from report to report. We have no influence which color the application takes for which bar.
- The time report of a previous sprint may change from report to report. We generate the bar charts at day of sprint end. If a task of a previous sprint is not finished, we will finish it during a later sprint and still use the "Sprint XY" tag of the previous sprint.

14.3.1 Final Time Reports

Up to and including the end day of the Term Project (December 20, 2024), we have worked a total of **690.13 hours** on the project. Therefore, we are **29.87 hours** under the predefined time.

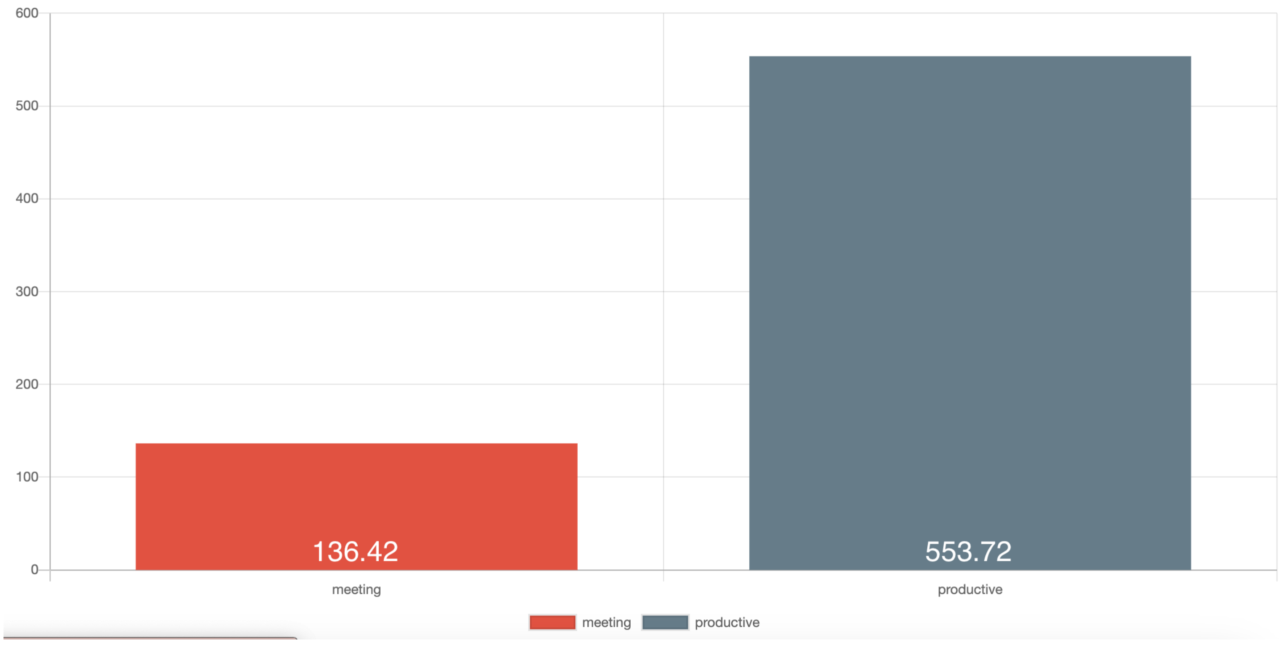


Figure 14.1: 690.13 hours divided by meeting and productive hours

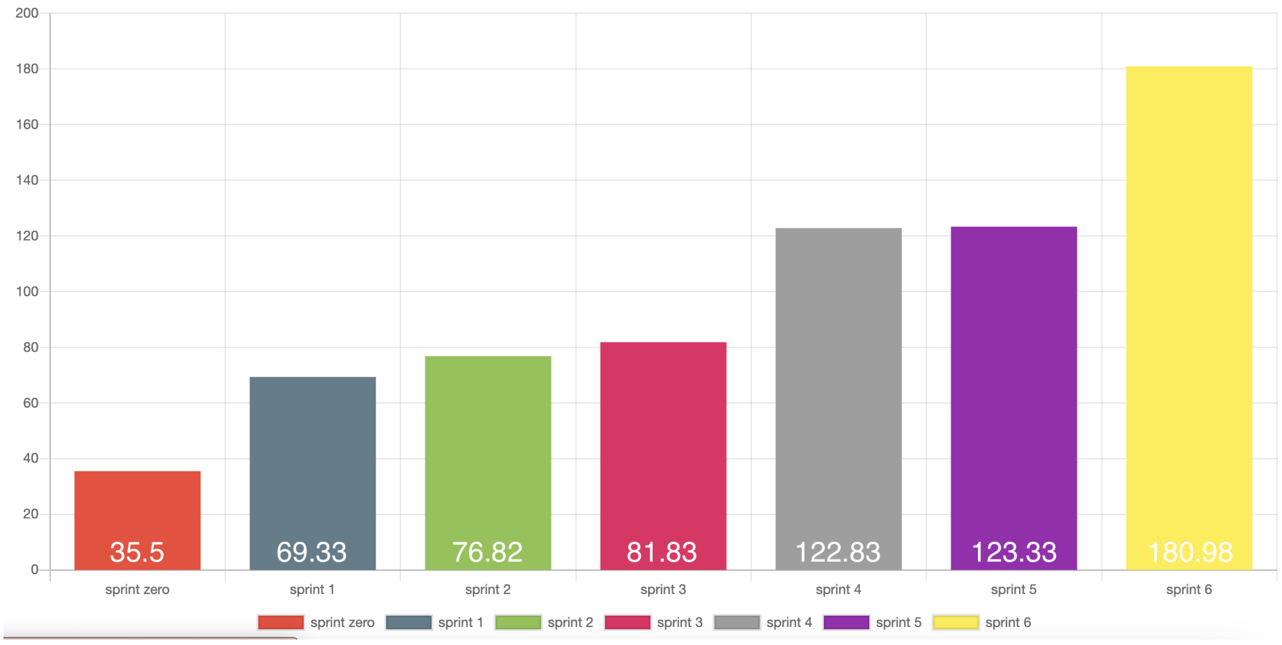


Figure 14.2: 690.13 hours divided by Sprints

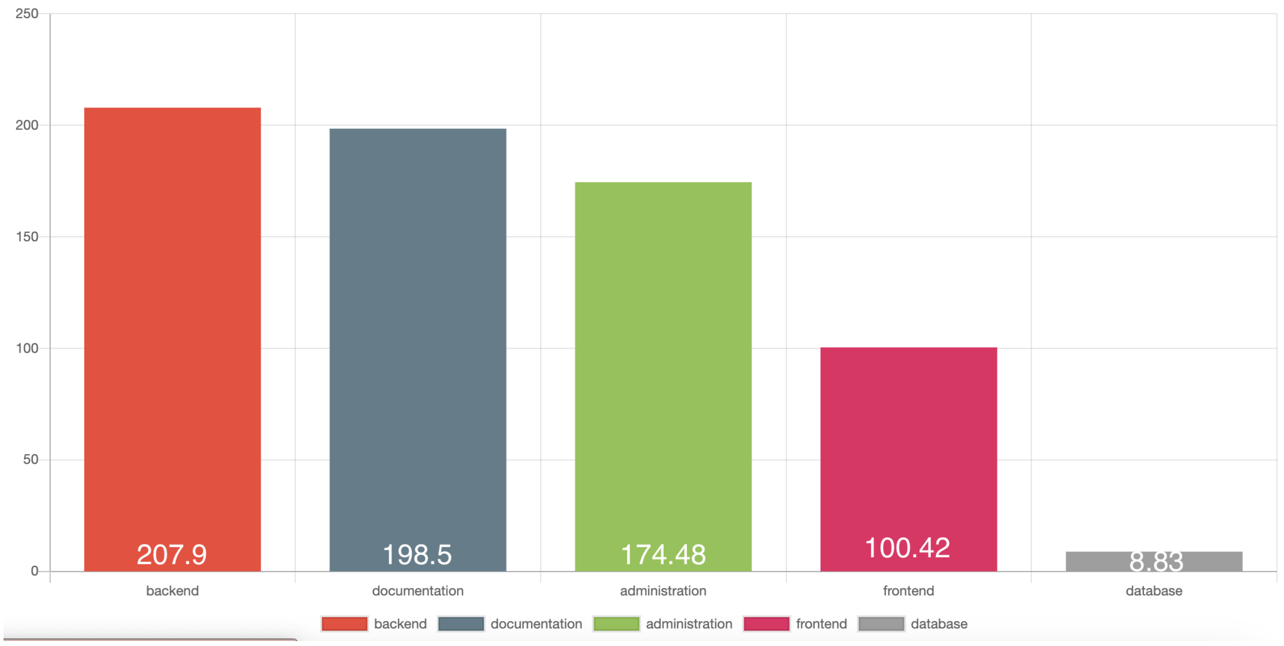


Figure 14.3: 690.13 hours divided by work area

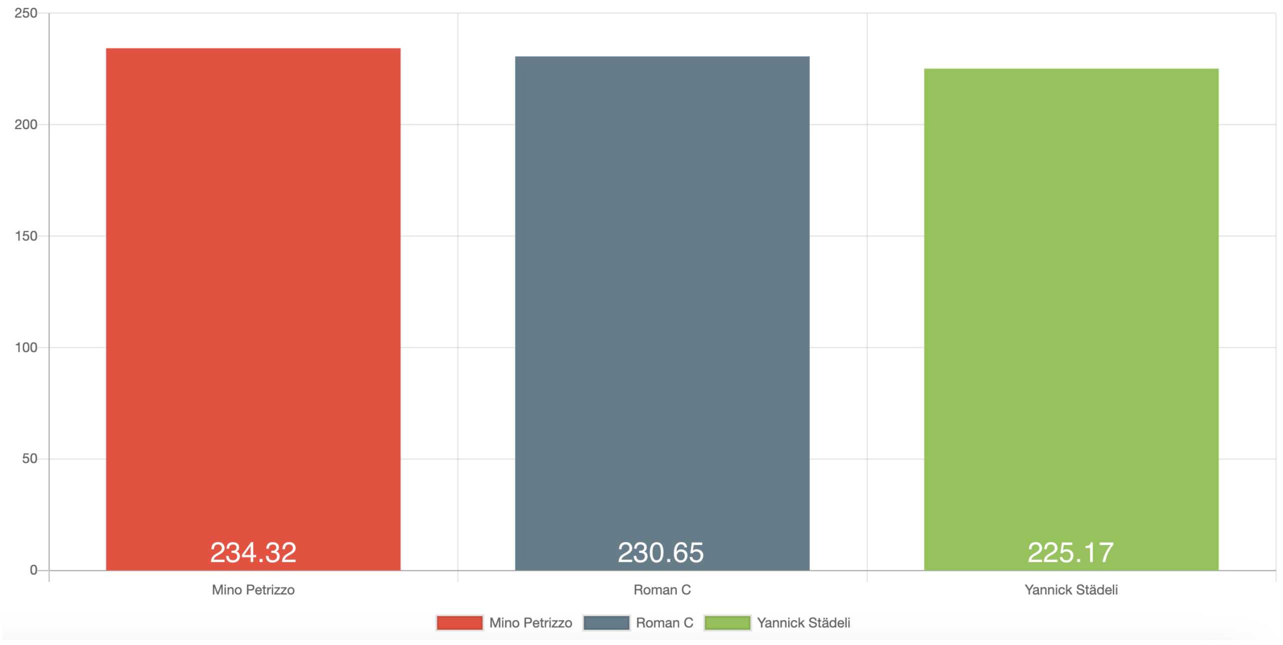


Figure 14.4: 690.13 hours divided by developer

14.4 Personal Reports

14.4.1 Report Yannick Staedeli

Similar to the experience I had during the SE project last semester, I learned a great deal through this term project. However, unlike the previous project, our team was smaller this time, which gave me the opportunity to take on more responsibilities across different areas. I felt that this team dynamic worked very well for me. Both of my teammates were extremely supportive and demonstrated advanced programming skills, which allowed me to benefit significantly from their expertise.

Coming from a networking background with little to no software knowledge before starting my studies, this project introduced me to many new concepts once again. Both Svelte and Go were unfamiliar to me, though I had some prior experience with HTML, CSS, and JavaScript, which provided a foundation for frontend work. As a result, I primarily focused on the frontend but also contributed to backend adjustments and implementations as needed.

That said, I believe there was room for improvement in our overall planning. In the last six weeks of the project, we were significantly busier compared to the earlier weeks. From the beginning, I felt our planning could have been more effective. For example, we spent many hours defining the epics, which in hindsight could have been streamlined.

Another challenge we faced was working on an existing project. It took a considerable amount of time and effort to understand the structure of the project, including how the frontend and backend were connected and where to find specific files. This process was quite time-consuming and unfamiliar to me, as I had never worked with a pre-existing code-base before.

A particularly significant milestone for me was writing code that is now publicly accessible and, hopefully, will someday be used by a diverse range of users. This was a first for me and a highly rewarding experience.

14.4.2 Report Mino Petrizzo

Technical Journey

As an web developer with exposure to various programming languages, this project presented an exciting challenge. While I have worked with multiple technologies, Go was new to me. However, my familiarity with C made the transition to Go relatively smooth, as the languages share similar structural paradigms. Svelte, though previously encountered in another module, offered an opportunity to deepen my understanding and apply it in a more comprehensive project context.

Team Dynamics

My teammates were supportive and highly motivated to reach the goal of this project. With strong ideas and only minor disputes, we got along well and were always in agreement in the end.

Challenges and Opportunities

Project Understanding

Working on an existing project was particularly challenging. Understanding the project structure, frontend-backend connections, and locating specific files was time-consuming and unfamiliar. This process tested my adaptability and problem-solving skills. Also with do documentation the only way was to understand the code or question our supervisor.

Planning Limitations

Our project planning was a significant area for improvement. We spent extensive time defining epics, and the workload became increasingly intense in the last six weeks. In retrospect, we could have streamlined our planning process to be more efficient. I also was one whole week ill and could not work as much, as I intended.

Personal Milestone

A highlight of this project was creating code that is now publicly accessible. The prospect of potentially helping diverse users through our software was incredibly rewarding and motivating.

Key Takeaways

- Importance of team collaboration
- Value of continuous learning
- Need for more structured project planning

14.4.3 Report Roman Cvijanovic

While very short and fast paced, this project was incredibly interesting and a valuable experience. During this project I've learned that having a united team is very important. While the initial slow pace was worrying the progress was always steady and once everything was established and well-planned, the goal appeared reachable.

Nevertheless, proper planning has been the biggest miss in this endeavor. There is no reason to fault any person solely and I believe it was a combination of lack of direction and knowledge which contributed to a slow start. But we still managed to pull through and deliver a satisfying product.

Everything considered this was a great experience. From the Advisory to each team member everything was great. Communication and planing was always clear and regular meetings benefited in progressing correctly towards our desired goal

14.5 Scripts

Cataloge Github Repos

1_catalogeGithubRepos.py

Cataloge Gitlab Repos

1_catalogeGitlabRepos.py

Calculate Contributor Numbers

2_getContributerNumbers.py

Calculate Commits History

3_getCommitsHistory.py

Analyze Contributor Data

4_analyze_contributers_data.py

Analyze Commit Data

5_analyze_commits_data.py

Analyze Commits and Contributors Plotting

6_analyze_commits_and_contributers.py 7_analyze_contribs_and_commits.py

14.6 Meeting Minutes

Meeting 2024-09-16

Open Tasks

- Which language?
- Where do we put the Documentation Repo? -> gitlab.ost.ch
- What about Projectmanagement - Which tool?
  - Distribute positions
- Initial Task distribution: Set up
  - Projectmanagement tool
  - Documentation Repo
- Read through flatfeestack repo -
  - Define initial key elements to tackle -

Finished Tasks

Task Resolutions

- Which language: Enligsh
- Where do we put the Documentation Repo? -> gitlab.ost.ch
- What about Projectmanagement - Which tool? -> Jira
  - Time tracker: timetracker (haha)
  - Distribute positions
    - developer
      - all
    - scrum master
      - Roman
    - product owner
      - Thomas
    - Team leader
      - Thomas
    - Tester
      - Thomas?
      - Mino

Task delegation

- Read and understand the repo
  - all
- Jira Setup
  - Yannick, Mino
  - sa-arbeit-ffs < Roman lädt ein
- documentation repo
  - Roman
- brainstorm some ideas for the project
  - Mino

2024-09-19

Discussion points

- Distribute roles
- proper task distributions

Open Tasks

- proper task distributions
  - start flatfeestack and make it running (till next week)
  - Draw graphic of overview FFS and paste it some where
- multiplicator with fraud protection (long term)
- populate jira

Discussions

- future meetings on Tuesdays, freestyle during the BlCh exercise lessons

Meeting 2024-09-24

Open Tasks

- Documentation
  - clean doc from SE stuff
  - requirements in general
- Quality Measures
  - Version-Control Guidelines –> Branches
  - Code guidelines in general

Task Resolutions

- Documentation: example thesis in teams
- Quality Measures: . . .

Open tasks

- Thesis discussion + decide how we go about our project
- Discussion: Repo Check Analyzer
- Next step for project

Discussions

- Our thesis requries:
  - management summary
  - product documentation - technical documentation
    - Quality measures (KPIs)
    - requirements
    - architecture and domain models
    - Solution
  - project documentation
    - Resources?
    - Schedule - Phases, Iterations and Milestones?
    - Risk
    - time tables

Questions for Thomas

- What kind of documentation style does he want.

2024-10-07

Open Tasks

- Discussion “Fraud Protection Inception of trusted repos”
  - Next Steps
- Frontend Sketches
- Pointers in regards to Architecture

Discussion “Fraud Protection Inception of trusted repos”

- check when last fork was created

Frontend Sketches

- Change frontend login
  - add login for foundations (can be checked with database or with the software, just like with admin login)
  - sign ups for foundations are also seperate
- Multipliactor
  - search: add toggle key for multiplicator -> only shows when you are a foundation

Trusted Repos Sketches

- trusted symbol in repo overview missing
- add sort and filter functions
- add trust value to admin page

General questions

- What happens if a foundation wants to make one-time donations themselves? –> they can do one-time donations
- Foundation validations? –> Should not be a focus prio 2

Questions for Thomas

- Can a foundation donate too? –> Yes they can
  - If foundations can sponsor, should the payment history and multiplicator payment history be split?
- What about badges for foundations? –> out of scope

2024-10-10

agenda items

- retrospective
- discuss results sprint 1
  - Functional Requirements
  - Non-function Requirements
  - Draw sketches/correct them
  - Inception of trusted Repos techniques
  - Inception of fraud protection
  - Create initial Architecture
  - Environment Setup
- define tasks sprint 2

Sprint 1 finish

Functional Requirements (MP)

- done
- but one task is still open, but question was defined

Non-function Requirements (RC)

- not yet started
- some ideas are floating around reliant on my main task fraud protection

Draw sketches/correct them (YS)

- done
- Write in documentation?

Inception of trusted Repos techniques (RC)

- Started with base work - documentation missing

Inception of fraud protection (YS)

- Worked on “fraud prevention for untrusted repos”

Create initial Architecture (MP)

- Created C1/C2, neatly documented
- Added use case diagramm

Environment Setup (RC)

- Split documentation in two parts, still missing make command to build both
  - Should be done by tomorrow

Questions for Thomas

- Does a foundation see the multiplicator of another foundation on a specific repo? -> Yes all open
- Does a sponsor see the multiplicator of a foundation on a specific repo? -> Yes all open

Retrospective

- MP
  - wants to start programming - he is dissatisfied with so much documentation stuff
  - all is good and well
  - good team work (except)
  - nothing to criticise as of now
- RC
  - All in all positiv outlook
  - good team and team effort
  - overestimated the initial tasks thus slight disappointment in slow progress - but that's a given, it's the inception phase of the project and understanding the real scope of our task
  - Outlook very positiv
- YS
  - positiv outlook
  - documentation helps to understand project better
  - better overview of what we want and need
  - our information sources are greatly increased
    * for example asking an external source like Ivan Buetler for security oriented inputs
  - All in all positive

Sprint 2 start

- see jira

2024-10-14

agenda items

- retrospective
- discuss progress sprint 2
  - Functional Requirements
- refine tasks sprint 2

Discussion with Thomas

- Update current progress
  - implement approval system -> approve that the
  - show risks, impact and likelihood
  - strategy for untrusted repos
- yannick discussed a good fraud prevention solution -> will share results later
- discussed open questions

Retrospective

Progress Sprint 2

2024-10-24

agenda items

- retrospective
- discuss progress sprint 2
  - inception trusted repos techniques
  - inception of fraud protection
  - repository metrics analysis
  - non functional requirements
  - turn sketches into frames
  - c1 & c2 of existing project with c3 for multiplier
  - login inputs
  - quality measures
    - mino done
    - yannick should be done close by end of sprint 2
- refine tasks sprint 2
- New task definitions

Retrospective

- RC
  - should book times better < done
  - really good progress overall
  - highly optimistic about the results
  - very satisfied with the frontend mock ups
  - team work is very good
- MP
  - mino is damned to work together with a faggot
  - very optimistic of the trust value - with consideration of automated trusted repos
  - supportive to each other
  - great teamwork
  - very satisfied of our progress
  - To not be tight on time: we should "hurry up" a bit

    - Simply said: no relaxation stay focused and on time (but don’t kill yourself over it)

- YS
  - very satisfied with the team work
  - the different discussions are refreshing and good - supplies good ideas
  - mino and yannick should take up more tasks or “responsibilities” to get on track with time management
    - but maybe that’s also just a
  - documentation is looking good and developing good

Progress Sprint 2

- inception trusted repos techniques
- inception of fraud protection
- repository metrics analysis
- non functional requirements
- turn sketches into frames
- c1 & c2 of existing project with c3 for multiplier
- login inputs

Start Sprint 3

- Tasks distributed
  - See jira

2024-10-28

agenda items

- discuss testing
  - test coverage

2024-10-29

agenda items

- discuss paper format (margin)
- was ist mit dem zitieren –> etwas unsicher wann muss man zititeren

2024-11-07

agenda items

- retrospective
- discuss progress sprint 3
  - inception trusted repos techniques
  - metric analysis
  - multiplier selection option
  - create c3 for multiplier
  - fraud protection refinement
  - trusted value backends
  - login from frontend not possible
- General: timeinvestment

Retrospective

- RC
  - team work and spirit still very good
  - a bit out of loop of what everyone is doing
  - progress seems to have slowed down a bit
    - Worried, we aren't spending too much time
    - optimism waning and getting nervous
- YS
  - Not bad
  - What was developed was good
  - team work is great
  - optimistic in regards to project
  - hours are "bad" we should be at 50% but we only have 30%
    - hope that we will manage it
- MP
  - happy to develope, not only stuck with documentation
  - trust value good that we see progress
    - would be happy if done with definition, but will be done soon

  - good team work, happy to work with everyone especially with himself
  - same issues with hours as yannick mentioned
  - if we don’t get lacks, we should be able to proceed smoothly
  - optimistic outlook

General: time investement

- on average every team member needs to spend another 8 hours every week to the project to meet the actually target time of 240 hours
- realistically, even if 4 additional hours are spent, it’s fine

Progress sprint 3

inception trusted repos techniques

- simply done, something from sprint 2 taken over to sprint 3

metric analysis

- roman:
  - done with his part
- mino:
  - definition sent, should be done? pending for review
    * added input from thomas
    * sql queries present
- will be transfered to sprint 4 (as intended)

multiplier selection option

- done, but a review tasks needs to be opened in sprint 4

create c3 for multiplier

- low priority, moved to print 4

fraud protection refinement

- more or less done, refinement will be done as part of general documentation
- closein sprint 3

trusted value backends

- nothing done, mosed to sprint 4

login from frontend not possible

- done by thomas

Closure sprint 3

- all open tasks are to be transfered to sprint 4
- missing times should be slowly invested in the new sprints

Start sprint 4

- see jira

2024-11-21

agenda items

- Retrospective
- Sprint 4 Conclusion
  - Create C3 for multiplier
  - Trusted value Backend
  - Trusted Repos: Frontend Visualization
  - Multiplier Activation
  - Review Metrics Analysis
  - Trust Value: Frontend Representation
  - Write tests for FFS-72 changes
- Sprint 5 Planing
  - 2 Weeks
- Sprint 6 Planing
  - 1 Full Week
  - Abstract
  - Documentation
  - Project Closure

Retrospective

- RC
  - Great Progress
  - A lot accomplished and done
  - Trusted Value Backend
    * mino great effort
  - very great team work and dynamic
  - very positiv outlook towards closure of project
- YS
  - very good progress in this sprint
  - happy to close this project as successfully as possible

    - team work is very good
        * less with roman but more with mino
    - Weekend work always available and flexible

- MP
    - very content to have a proper solution in mind
    - team work is great
    - worked around everywhere thus has overview over the whole progress
    - confident that we are steering according to plan and that in the last week only bug fixes and code sanitization is required

Sprint 4 Conclusion

- Move task FFS-97 to Sprint 4 and close it?
- Move task FFS-98 to sprint 4 and close it?
- Move User Story FFS-105 to sprint 4 and close it?
    - We can't really accomodate every single low res monitor.
    - Full HD is fairly standard today and this should be regarded as a minimum requirement.

Create C3 for multiplier

- initial draw done
    - subsequent changes requried according to changes of project
    - moved to backlog

Trusted value Backend

- mostly done
    - refinement required
    - review required

Trusted Repos: Frontend Visualization

- done
- handle what happens when a page reload happens or moving to another page
- review requried

Multiplier Activation

Review Metrics Analysis

Trust Value: Frontend Representation

Write tests for FFS-72 changes

- done?

Sprint 5

- see jira

Sprint 6

- see jira

2024-12-03

Agenda Talk with thoams

- Repo Assessments
  - satisfied with presentation
    - any changes
    - colors?
    - presentation?
    - design?
    - additional information?
  - repo analysis breakdown
  - forced analysis
- Multiplier Calculation
  - WIP
- Main Merge?
  - how intended?
  - simply leave everything into SA branch?
  - delete?
- Threshold values
- General
  - BA - themenvorschlag

Thomas

- rewritten auth completely: no password login anymore
- main should run again
- will take a look how likely it is to merge SA into main

Repo Assessment

- showcased
- redundant information
- tables and partial health values

    - table create total value
    - overall OK

- Forced Analysis Question: How doesn frontend know, when the analysis is done?
    - aa

BA Themenvorschlag

- über metamask
- DAO - forum
    - sign in with
    - is it that use?
    - does the user have the nft
    - only
- yannick:
    - andere arbeiten von thomas:
        * maybe flatfeestack
        * Transportprotokoll alternative QUIC
    - kubernetes cloud deployment
        * costs expload
        * digital ocean 5 services - up to USD 70/dollars
        * kubernetes significantlly more expensive
- thomas:
    - instead of kubernetes:
        * write some kind of abstraction to let everything run on one host

Agenda

- Besprechung repo weight
    - fortgeschrittener als ganzzeitlich bekannt
    -
- Besprechung SA
    - arbeitesverteilung

    * RC
        - Dokumentation - aufräumen
    * YS
    * MC

- Besprechung BA Themen
  - DAO Forum Integration (into flatfeestack)
  - Kubernetes Deployment
    * Scope nicht komplett BA
    * Integration von Flatfeestack in Deployment Szenario -> DeployTeam -> Continuous Deployment -> Testing auf eigene Instanzen
  - Transferprotokoll Alternative zu QUIC (TomTP https://github.com/tbocek/tomtp)
    * sounds interesting
- button health value 0
- press on it to reanalyse
-