FINGRO

Arshdeep Singh(S224750073)
Emmalu joseph(S224791713)
Shrikesh(S224404506)
Umar Khan(S223744692)

Software Requirements Specification (SRS) for FinGro

Project Name: FinGro (Finance & Grocery Planner)

1. Introduction

1.1 Purpose

The purpose of the FinGro web application is to assist users in managing their grocery planning and financial tracking efficiently. By providing an integrated platform for meal planning, grocery list generation, expense categorization, and savings monitoring, FinGro empowers users to make informed financial and lifestyle decisions.

1.2 Scope

FinGro serves as a comprehensive web-based solution for both meal and financial planning. It supports the following key functionalities:

- **Grocery Planning:** Allows users to plan their meals and generate consolidated grocery lists based on family size and selected dishes.
- **Expense Tracking:** Enables users to track daily, weekly, and monthly expenses, categorized for easy review.
- **Savings Insights:** Users can set financial goals and visualize their progress toward achieving them.

FinGro is designed to operate on desktop and mobile web browsers and supports various user groups, including individuals, students, and families.

1.3 Definitions and Abbreviations

- **SRS:** Software Requirements Specification
- **UI:** User Interface
- **API:** Application Programming Interface
- **GDPR:** General Data Protection Regulation (data privacy regulation)
- **MongoDB:** A NoSQL database used for data storage

2. System Overview

FinGro functions as a standalone web application built using modern web technologies. The front end is developed using HTML, CSS, JavaScript, and Materialize CSS, while the back end runs on Node.js and MongoDB for secure and scalable data storage.

3. Functional Requirements

3.1 Grocery Planning Module

Description: Allows users to plan meals and generate weekly grocery lists.

Requirements:

- Users can add or remove dishes for each meal of the week (breakfast, lunch, dinner).
- The system calculates grocery quantities based on serving sizes.
- The module fetches recipes and ingredients from external APIs if required.

User Story:
"As a user, I want to plan my meals and generate a comprehensive grocery list for the week to simplify my shopping."

3.2 Expense Tracker Module

Description: Tracks user expenses and categorizes them for budgeting and analysis.

Requirements:

- Users can add expenses with descriptions and categories (e.g., groceries, rent, utilities).
- The system generates summaries of daily, weekly, and monthly expenses.
- Visual reports (e.g., bar charts, pie charts) display expense breakdowns using external charting libraries.

User Story:
"As a user, I want to track my monthly expenses so that I can monitor my spending and plan my budget accordingly."

3.3 Savings Goals Module

Description: Provides users with tools to set and monitor financial goals.

Requirements:

- Users can set savings goals with specific amounts and timelines.
- The module displays progress toward each goal through graphical insights.
- Notifications alert users when they are close to meeting or missing their goal targets.

4. Non-Functional Requirements (NFR)

- **Performance:** All user actions (e.g., adding expenses, generating grocery lists) should complete within 3 seconds.
- **Security:** All user data must be encrypted both in transit (HTTPS) and at rest (MongoDB).
- **Scalability:** The system should handle up to 100,000 concurrent users.
- **Usability:** The UI must be intuitive and easy to use, even for non-technical users.
- **Reliability:** The application should maintain 99.9% uptime, ensuring availability.

5. System Architecture

5.1 High-Level Overview

- **Frontend:** Developed using HTML, CSS, JavaScript, and Materialize CSS for responsive styling.
- **Backend:** Built using Node.js for server-side logic and MongoDB for database storage.

5.2 Data Flow

- **Inputs:** User-provided data, such as meal preferences and expense details.
- **Processes:** Data aggregation, API calls for recipes, and financial calculations.
- **Outputs:** Consolidated grocery lists and visualized expense summaries.

6. External Interfaces

- **User Interface:** Web application accessible via browser with navigation options for meal planning and expense tracking.
- **API Integration:** External APIs for fetching recipes and ingredient data.
- **Communication Interface:** HTTP/HTTPS protocols for client-server communication.

7. Constraints and Assumptions

- **Assumptions:**
    - Users have a stable internet connection while using the app.
    - MongoDB is used as the primary database for storage.
- **Constraints:**
    - The web application will support only English language for the initial release.

8. User Stories and Use Cases

8.1 Grocery Planner Use Case

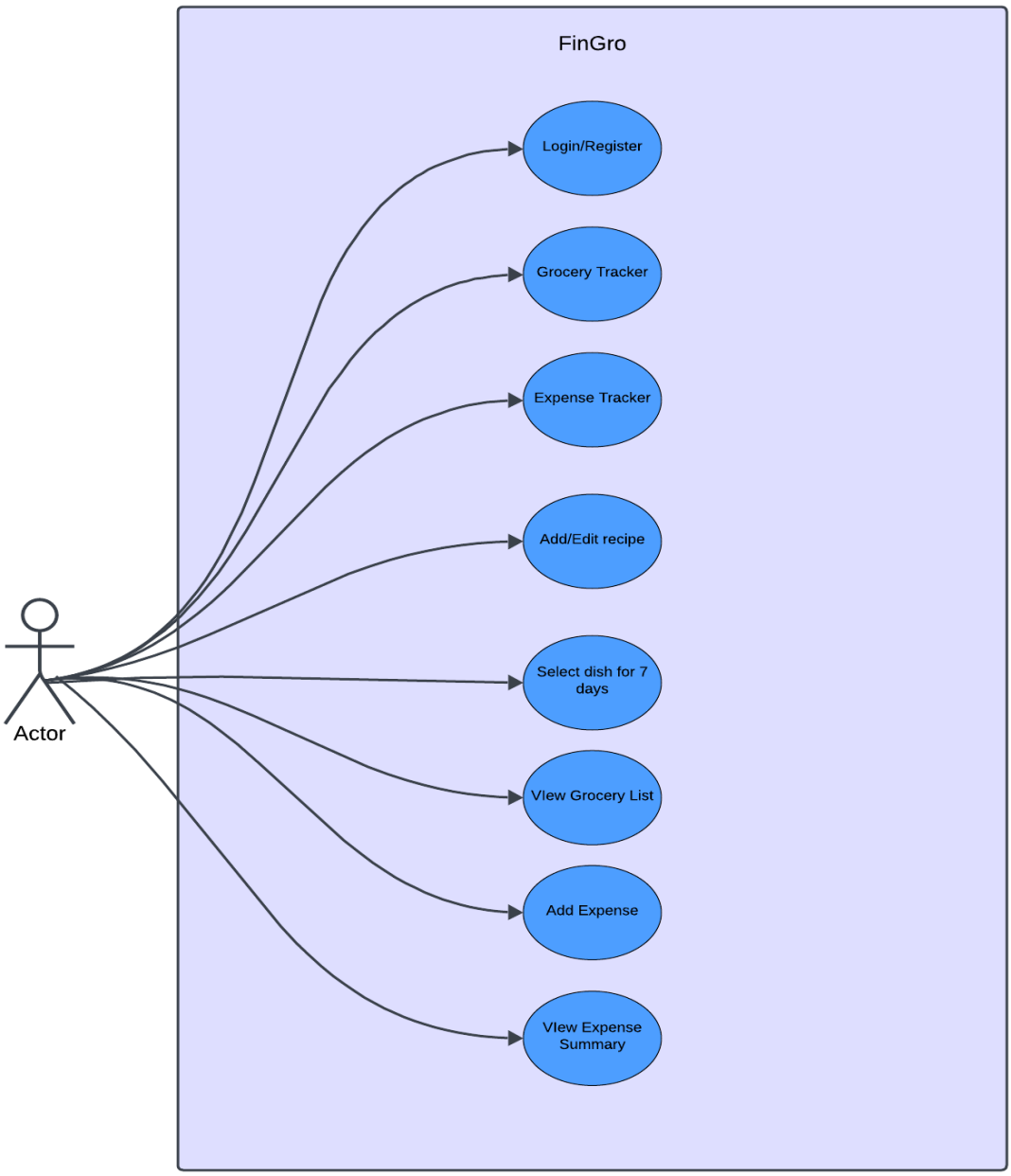
Scenario: A user plans meals for the week and generates a grocery list.
Steps:

1. Log in and select the "Grocery Planner" section.
2. Add dishes for each meal of the week.
3. The system generates a grocery list with ingredient quantities.

8.2 Expense Tracker Use Case

Scenario: A user tracks expenses for the month.
Steps:

1. Log in and navigate to the "Expense Tracker" tab.
2. Add expenses and categorize them (e.g., "Groceries" or "Transportation").
3. The system displays a summary report of total expenses per category.



9. Work Breakdown Structure (WBS)

- **Frontend Development:** Implement the UI for meal planning and expense tracking.
- **Backend Development:** Create REST APIs, connect MongoDB, and implement user authentication.
- **Testing:** Conduct usability tests and load tests for performance evaluation.
- **Deployment:** Finalize and host the web app on a cloud platform (e.g., AWS).

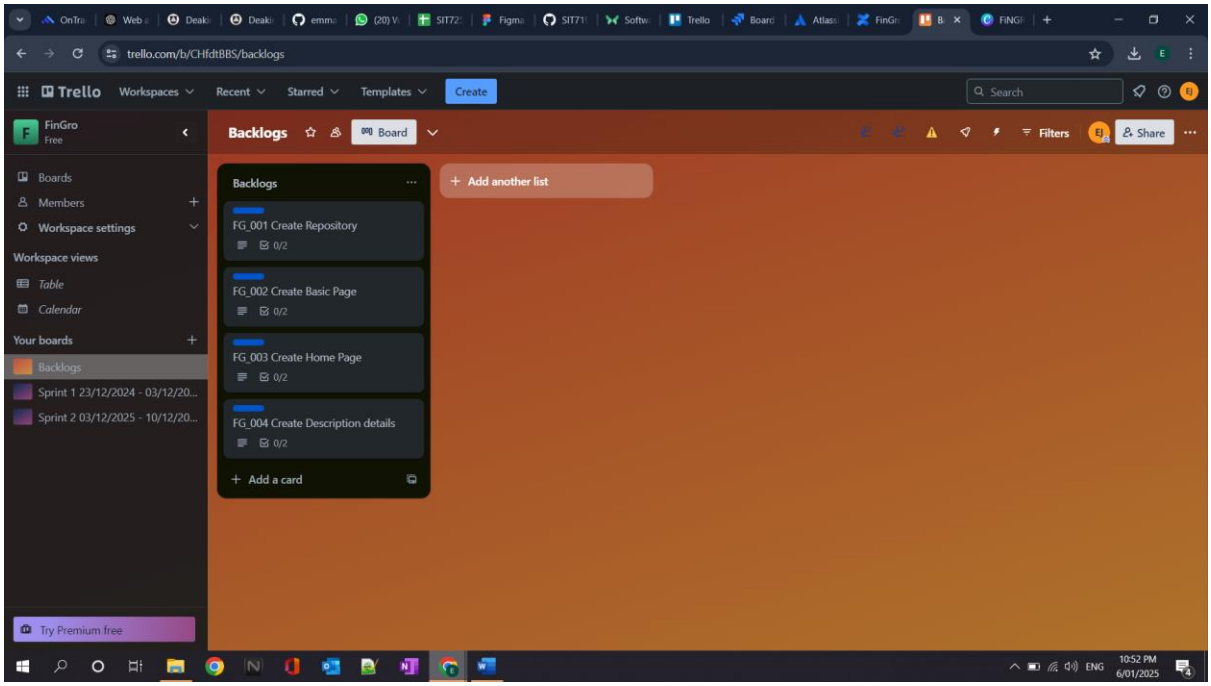
10. Approval and Sign-Off

This document must be reviewed and signed off by all stakeholders, including team members, project managers, and product owners, before proceeding with development.

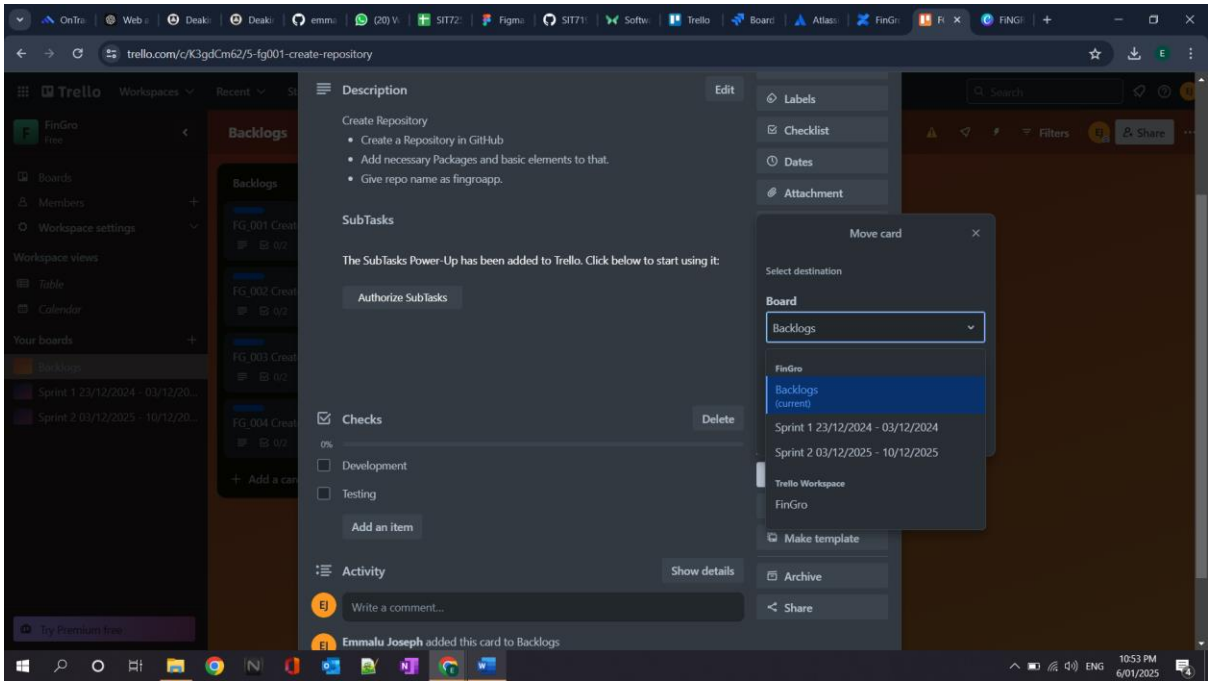
Trello Screen Shots

Trello Link : https://trello.com/b/CHfdtBBS

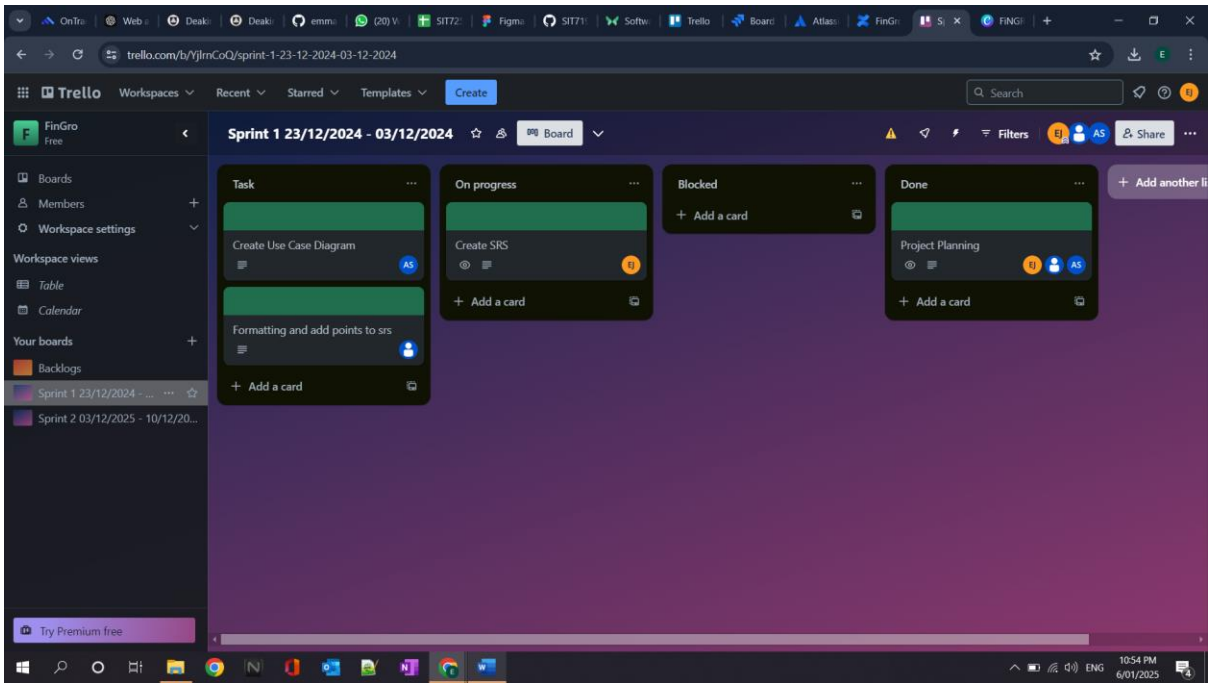
1. The overall Trello board structure used for managing tasks during the project.

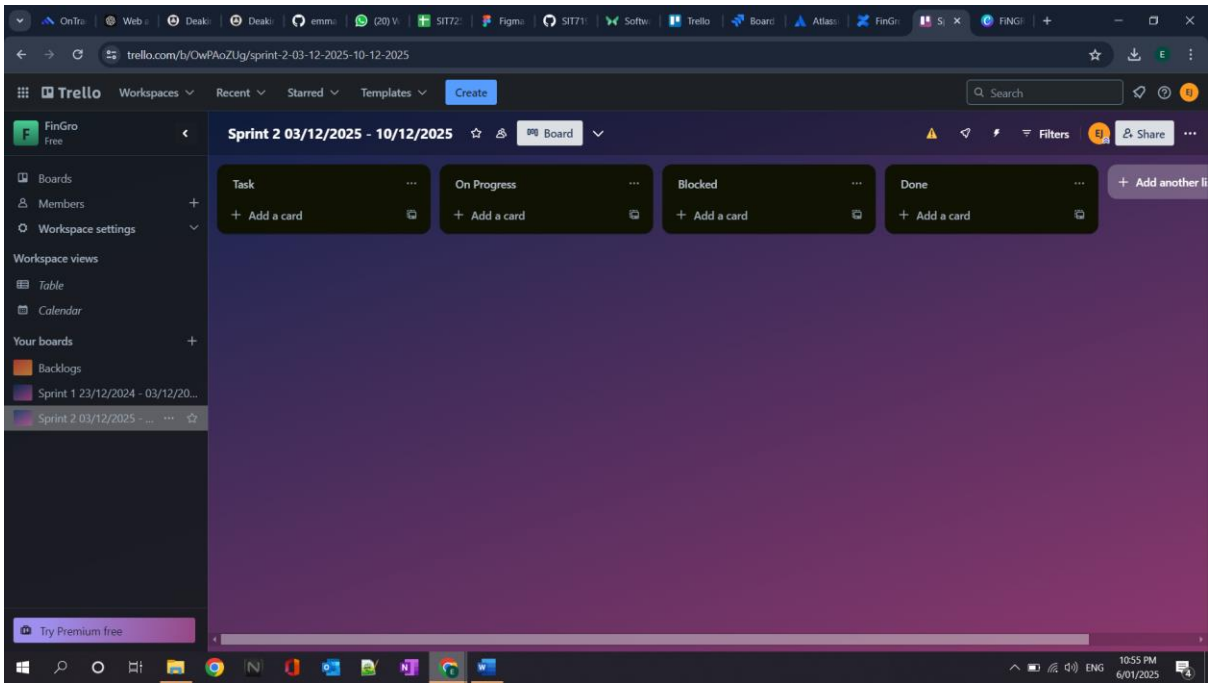


2. From This we move each task to the corresponding sprint.

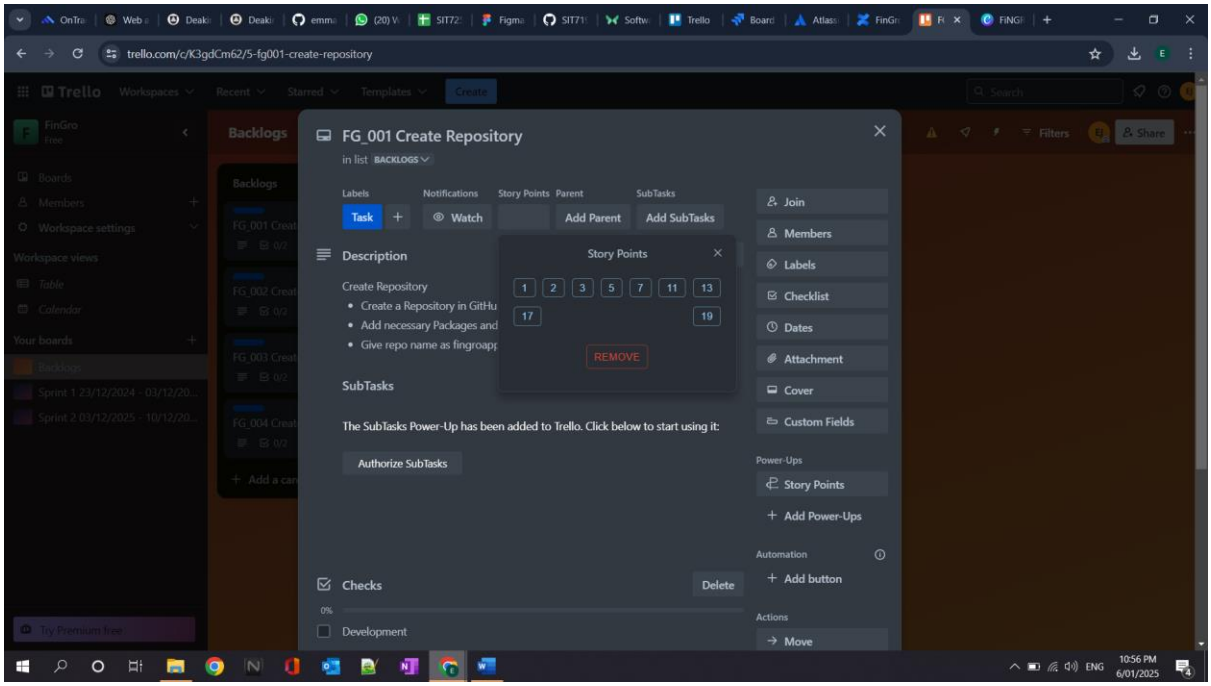


3. In Each Sprint We have 4 stages for a task.

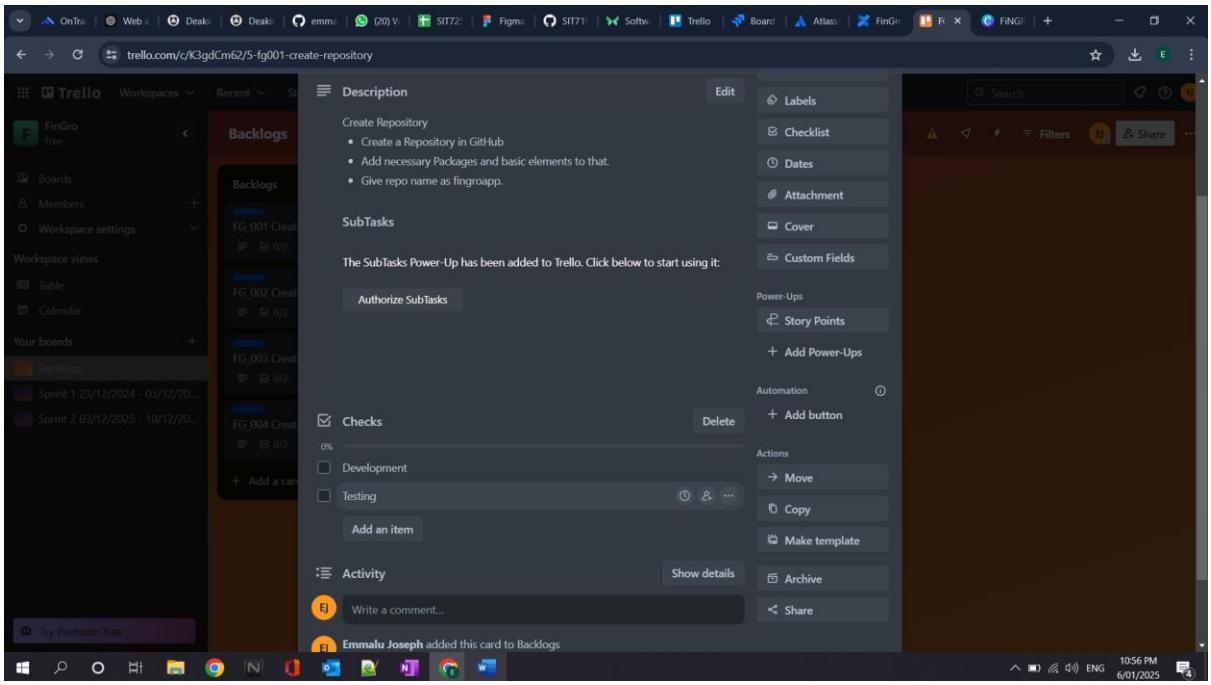




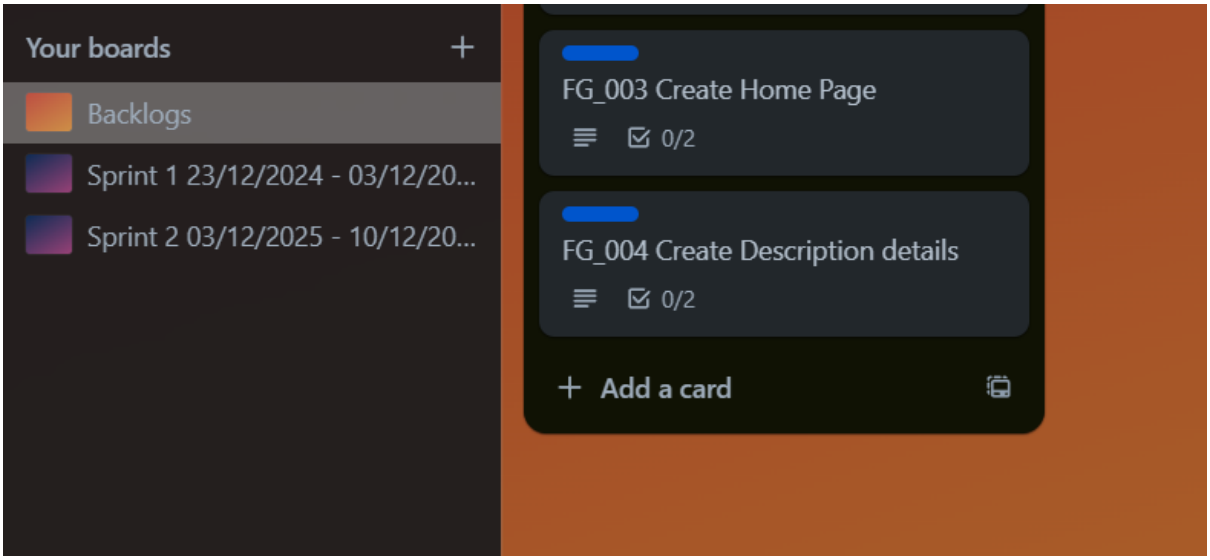
4. Each Task has story points. Story points mean hours needed to complete that task



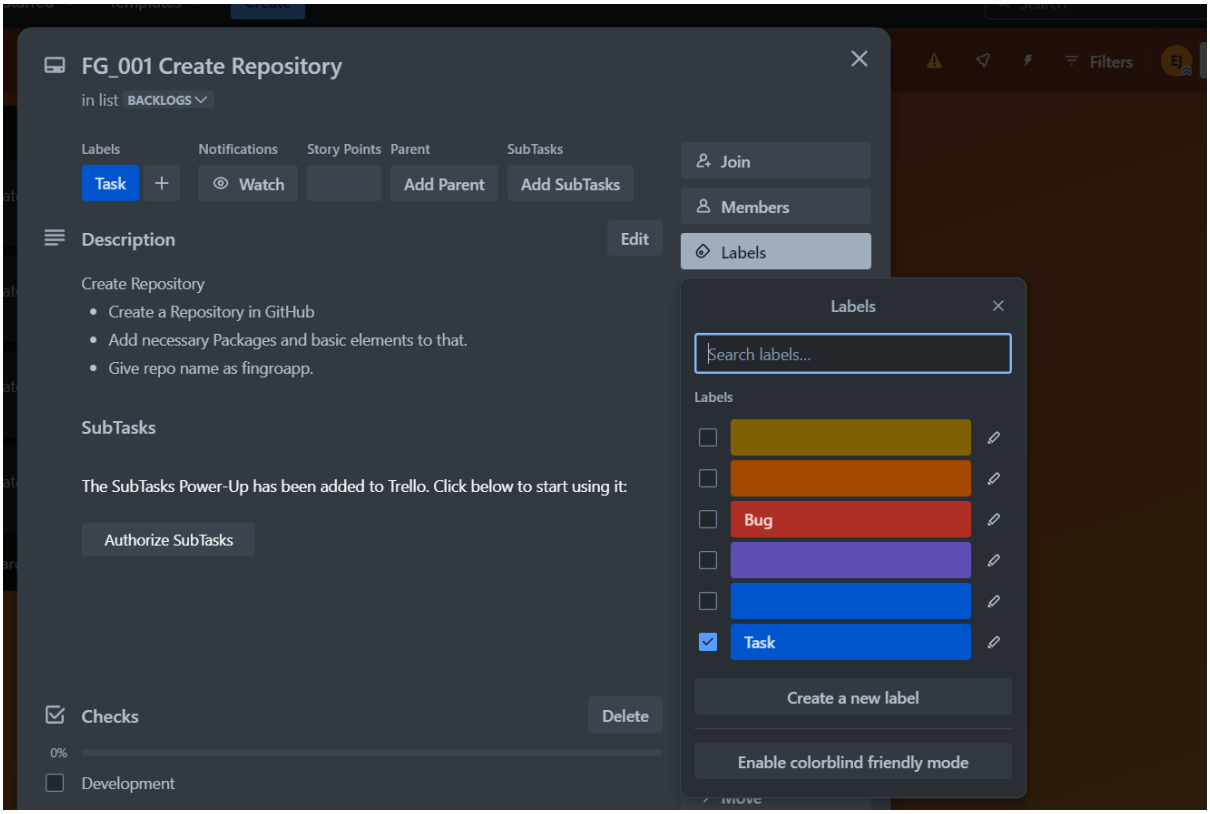
5. Each Task has the check list.



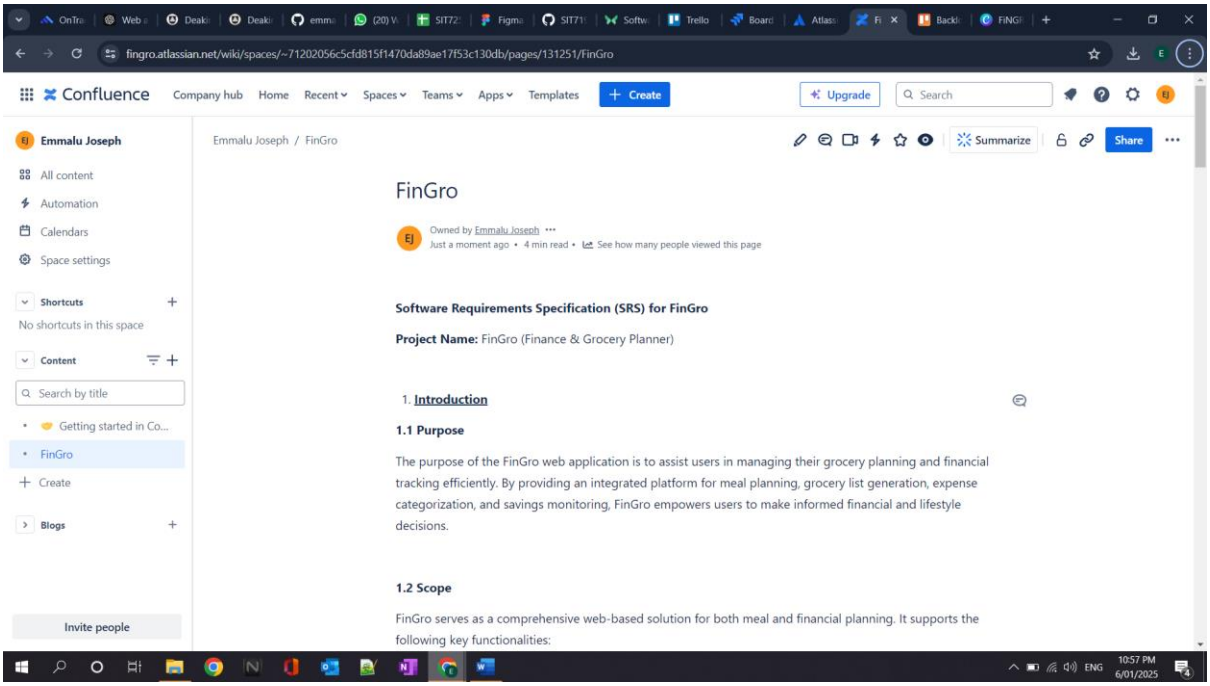
6. Boards created



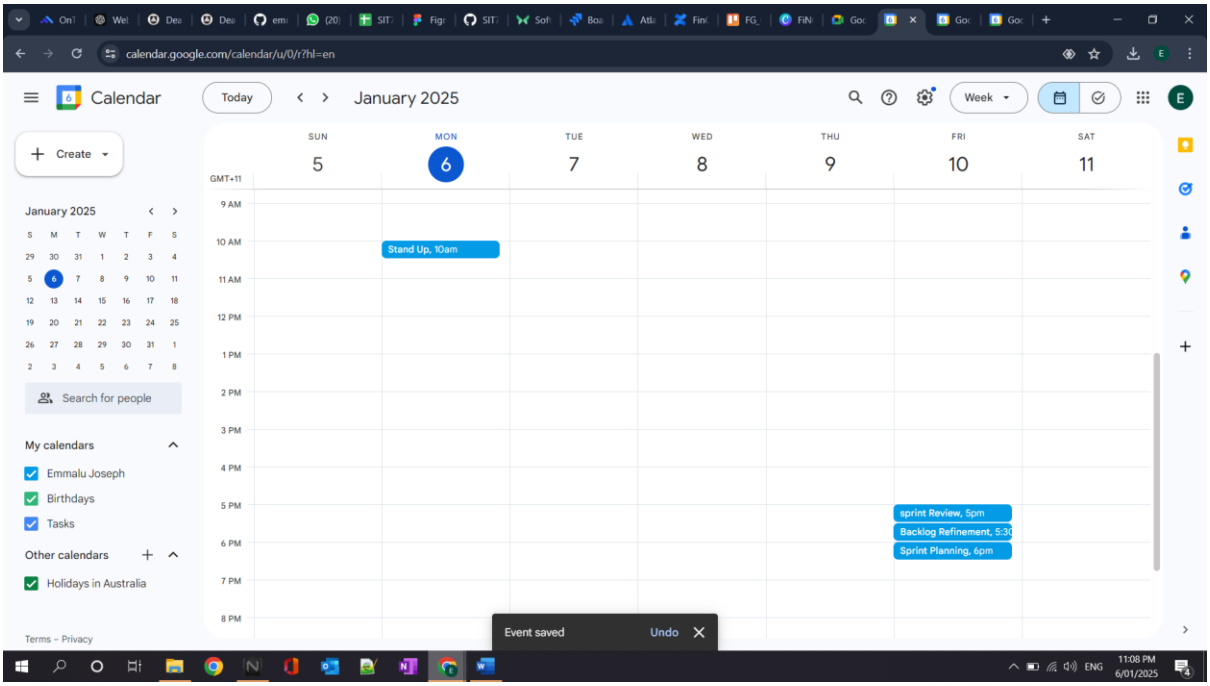
7. Each ticket has different labels according to the functionality.



8. We are using Confluence page for documentation.



9. Meetings Scheduled on Each sprint



Reference:

Diceus. (n.d.). How to Write an Effective Software Requirements Specification (SRS)
Document. Retrieved from [https://diceus.com/custom-software-requirements-specification/].