CS673 Software Engineering
Team 1 - trackr
Project Proposal and Planning

Team Member	Role(s)	Signature	Date
Timothy Flucker	Design / Implementation Leader	*Timothy Flucker*	05/10/2022
Jean Dorancy	Team Leader	*Jean Dorancy*	05/11/2022
Xiaobing Hou	Security Leader	*Xiaobing Hou*	05/11/2022
Weijie Liang	QA Leader	*Weijie Liang*	05/13/2022

Revision history

Version	**Author**	**Date**	**Change**
0.1	**Timothy Flucker**	**05/12/2022**	**Adding content for assigned sections (1, 2, 3, 6b)**
0.2	**Jean Dorancy**	**05/13/2022**	**Code Review Process**
0.3	**Xiaobing Hou**	**05/13/2022**	**Configuration Management Plan**
0.4	**Weijie Liang**	**05/13/2022**	**Quality Assurance Plan**

0.5	**Jean Dorancy**	**05/14/2022**	**Management Plan and other edits**
0.6	**Weijie Liang**	**05/14/2022**	**Add details in Quality Assurance Plan**
0.7	**Jean Dorancy**	**05/15/2022**	**Risk Management**
0.8	**Jean Dorancy**	**05/16/2022**	**Minor Edits**

1. Overview

This project is directly inspired from financial services such as Mint and TrueBill, namely software that tracks transactions against a bank account. The purpose of this project is to

have an informal way for users to enter information related to a bank account and transactions against it to understand their spending behavior and the amount of money deposited and withdrawn from their account.

A user will create an account for themselves which will provide them with credentials used for authorization and authentication for the other APIs. Once an account is created, the user will be able to enter in bank account information and then start to enter transaction information against that account. Once this information is entered into the system the user will be able to understand their spending behavior relative to that account.
This project will be developed using Java 1.8 with the Spring Boot Framework, and developed using an IDE such as IntelliJ. Version control of the project will be handled by Git and the class organization GitHub account. An embedded H2 database will be used to contain an initial data set for the application as well as store any information entered by the user after they run the application, however this may be modified to use a PostgreSQL database later in development. The application will be deployed using the Heroku platform.

2. Related Work

This application is based on applications such as Mint and Trubill that are used to help users view and understand how they spend their money. These applications are built by large financial institutions which provide users with many resources and features to help understand their spending habits, however it also floods them with ads for new bank accounts and credit cards which can be detrimental.

Our application is a much simpler approach to tracking spending behavior and only allows users to input data and view their transactions relative to their bank accounts.

3. Proposed High level Requirements

a. Functional Requirements
   (For each functional requirement, please give a feature title and a brief description using the following format: As (a role), I want to (action), so that (value).)
   i. Essential Features (the core features that you definitely need to finish):
      (For each essential features, please give a rough estimation in terms of person hours or an range of person hours)
   ii. Desirable Features (the nice features that you really want to have too):
   iii. Optional Features (additional cool features that you want to have if there is time):
   iv. Existing Features (delete this item if your project starts from scratch):

b. Nonfunctional Requirements
   i. Security requirements

For the following features listed below, **point values are used to indicate the relative complexity of each feature**. Lower values indicate a feature that is easier to implement and larger point values

indicate a complex story that will generally require more time and effort to fully implement. The scale for these point values is from 1 - 8 using a Fibonacci sequence (1, 2, 3, 5, 8) which is used in some Agile-Scrum projects.

- Functional Requirements
    - Essential features
        - User Management
            - Title: Create User API
                - Description: As a customer, I want to be able to create a user account for the application, so that I can have my data associated with my account.
                - Priority: 0 - Critical
                - Points: 3
                - Estimate: 2 - 4 hours
        - Bank Account Management
            - Title: Create New Bank Account API
                - Description: As a user, I want to create a bank account record so that I can track deposits and withdrawal transactions against it.
                - Priority: 0 - Critical
                - Points: 3
                - Estimate: 2 - 4 hours
            - Title: Modify Bank Account API
                - Description: As a user, I want to modify a bank account record so that I can update its relevant information to be current.
                - Priority: 1 - High
                - Points: 3
                - Estimate: 2 - 4 hours
            - Title: Deactivate Bank Account API
                - Description: As a user I want to be able to deactivate a bank account record so that I no longer see that data.
                - Priority: 2 - Medium
                - Points: 3
                - Estimate: 2 - 3 hours
            - Title: View All of my Bank Account
                - Description: As a user, I want to be able to view all of my bank accounts, so that I can interact with all of the data I have entered.
                - Priority: 1 - High
                - Points: 3
                - Estimate: 2 - 3 hours
            - Title: View Specific Bank Account By ID

        - Description: As a user, I want to be able to view a specific bank account using a unique identifier, so that I view its data and take any necessary action against it.
        - Priority: 1 - High
        - Points: 3
        - Estimate: 2 - 3 hours
    - Transaction Management
        - Title: Create New Transaction API
            - Description: As a user, I want to create a transaction record linked to a bank account so that my bank account information is up to date.
            - Priority: 0 - Critical
            - Points: 3
            - Estimate: 2 - 4 hours
        - Title: Modify Transaction API
            - Description: As a user, I want to modify a transaction record so that I can update its relevant information to be current.
            - Priority: 1 - High
            - Points: 3
            - Estimate: 2 - 4 hours
        - Title: Void a Transaction API
            - Description: As a user I want to be able to void a transaction record so that I can reverse the transaction and update my bank account.
            - Priority: 2 - Medium
            - Points: 3
            - Estimate: 2 - 3 hours
        - Title: View All of my Transactions
            - Description: As a user, I want to be able to view all of my transactions against a bank account, so that I can see all of the activity of that bank account.
            - Priority: 1 - High
            - Points: 3
            - Estimate: 2 - 3 hours
        - Title: View Specific Transaction By ID
            - Description: As a user, I want to be able to view a specific transaction using a unique identifier, so that I view its data and take any necessary action against it.
            - Priority: 1 - High
            - Points: 3
            - Estimate: 2 - 3 hours
- Desirable Features
    - User Management

- Title: User Password Reset
    - Description: As a user, I want to be able to reset the password of my account so that, if I forget my password I do not lose access to my account.
    - Priority: 1 - High
    - Points: 3
    - Estimate: 4 - 8 hours

- Optional Features
    - Web GUI
        - Title: Create User Login page
            - Description: As a user, I want to have a web interface to log into the application so that I do not have to use the APIs.
            - Priority: 3 - Low
            - Points: 5
            - Estimate: 5 - 10 hours
        - Title: Create Home page
            - Description: As a user, I want to have a homepage that shows all of my relevant information so that I can interact with my data more easily.
            - Priority: 3 - Low
            - Points: 5
            - Estimate: 20 - 30 hours
        - Title: Add information Information pages
            - Description: As a user I want to have a page where I can add information such as my bank account, or transaction information so that new information can be added quickly and easily.
            - Priority: 3 - Low
            - Points: 8
            - Estimate: 20 - 30 hours
        - Title: Edit Information pages
            - Description: As a user I want to have a page where I can edit information such as my user, bank account, or transaction information so that I can keep my data up-to-date.
            - Priority: 3 - Low
            - Points: 8
            - Estimate: 20 - 30 hours
        - Title: Configure Session Management for web pages
            - Description: As a developer, I want to enable session management for all web pages so that the application is secured.
            - Priority: 0 - Critical
            - Points: 5

        - Estimage: 10 - 20 hours
- Nonfunctional Requirements
    - Title: Create a Swagger document for the APIs
        - Description: As a developer, I want a swagger document which shows how my APIs are designed, so that I have a relevant technical project artifact.
        - Priority: 0 - Critical
        - Points: 1
        - Estimate: 5 - 10 hours
    - Title: Authenticate API requests
        - Description: As a user I want all of my API requests to be authenticated using a basic authentication strategy so that my data is protected from random and unsolicited access.
        - Priority: 0 - Critical
        - Points: 5
        - Estimate: 5 - 10 hours
    - Title: Authorize API requests
        - Description: As a user I want only API requests I send or from a system admin to be authorized to access my data so that only I and a system admin can view my data.
        - Priority: 0 - Critical
        - Points: 5
        - Estimate: 5 - 10 hours

4. Management Plan

a. Objectives and Priorities

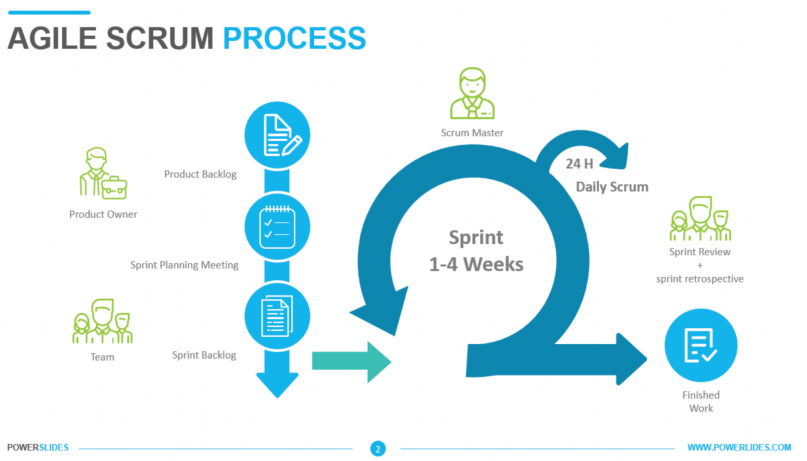
- Every student actively contributes and practices Scrum rituals.
- Opportunities for everyone to learn.
- Focus on best practices to maintain high quality.
- Bottom up unit-testing approach for all feature implementation.
- Complete all essential features without compromising software quality.
- Complete some desirable features.

b. Process Model

We are planning to use the Scrum framework as the process for developing this project. Based on student availability we are looking to implement all the scrum events.

- **Sprint**: We will have sprints of iteration length given the class short duration.
- **Daily Scrum**: Implemented virtually where each team member posts updates everyday between 7pm - 7:15pm.
    - What did you do yesterday?
    - What will you do today?

    - What blockers stand in your way?
- **Sprint Retro and Planning**: Briefly talk about the last iteration, estimate and decide on the goals for the sprint.



We only had two meetings for iteration 0. We are using PivotalTracker to manage our requirements. Everyone is expected to create user stories in the backlog which are discussed in the Sprint Planning.

c. Communication Plan

We are using **Zoom** for live classrooms hence it makes a natural choice for synchronous meetings. We have a **Discord** channel for instant messaging. In the team text channel we have two dedicated threads: A **Daily Scrum** thread where folks post updates every day between 7pm - 7:15pm. A **Pull Requests** thread that is used to post PR to be reviewed by the team. If we run into roadblocks, we post in the channel for help and if we can't find a solution then we reach out to the facilitator and lastly email the professor according to school policy.

d. Risk Management (need to be updated constantly)

The main risks we have identified for the project after filling the risk management sheet and ways to address them are as follows.

Risk Title	**Priority**	**Plan**
Constant requirements change	8	- Favor generic solutions that are adaptable - Embrace change and build the product using appropriate design patterns

		- Emphasize communication with stakeholders to ensure requirements are well understood
Unclear requirements	12	- Sprint review meeting with stakeholders - Constant communication with stakeholders - Stakeholders to review all requirements and answer questions.
Lack of motivation or responsibility	15	- Opportunities for everyone to learn - Everyone to work on things they are interested in - Practice the Scrum rituals which keeps the team engaged
Scope creep	16	- Meeting with stakeholders about increase of scope which might impact project schedule - Dropping some of essential features - Alternative technical solutions

Risk Management Sheet Link: here

e. Timeline

Sheet with detail estimates for each iteration: here

Iter	**Functional Requirements(Essential/Desirable/Option)**	**Tasks (Cross requirements tasks)**	**Estimated /real person hours**	**Links**
1	- User Management	- Architecture and design - Test plan	50	

	- Bank Account Management	- Spring Boot tutorial - Create a Spring Boot project - Deployment pipelines - DB schema for user - Implement create user - DB schema for bank account - Implement create new bank account - Implement modify bank account - Implement deactivate bank account - Implement view all my bank account - Implement view account by ID		
2	- Transaction Management	- Architecture and design - Test plan - Create pipeline to automatically run project tests on PR request - DB schema for transaction - Implement create new transaction - Implement modify transaction - Implement void a transaction - Implement view all of my transactions - Implement view transaction By ID	35	
3	- User Management - Create a Swagger document for the APIs - Web GUI	- Architecture and design - Test plan - Update user DB schema if needed - Implement user authentication and session management - Implement password reset - Implement authorization - Setup project with React and Wepack - Implement home page - Implement user login page - Implement create user page - Implement create and modify bank account page - Implement view bank account page - Implement view all my bank accounts page - Implement create and modify transaction page - Implement view transaction page - Implement view all my transactions page - Implement password reset page	165	

5. Configuration Management Plan

a. Tools

Dev Tools

- Java (1.8, Download here)
- Swagger (api design, online editor here)
- Lombok (java dependency, website to download and install here)
- Maven (java build tool / dependency management, download here)

Version Control Tools

- Git (Download here)
- GitHub (Sign up here and our project here)

IDE Tools

- IntelliJ (Download here)

Project Management Tools

- PivotalTracker (Our project here)
- Google Drive (Our project here)

Test Tools

- Postman (Sign up here)
- JUnit5 (documentation and how to use here)

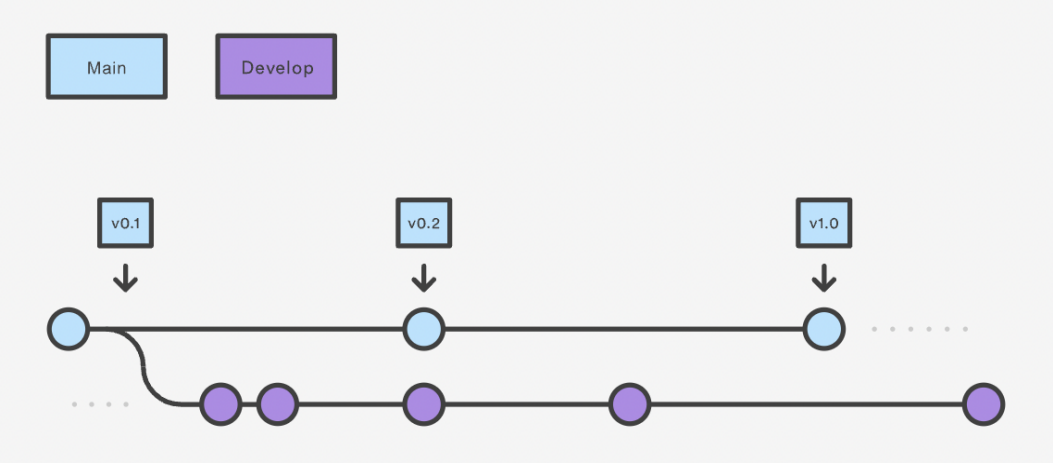
Meeting and Dissemination Tools

- Zoom
- Discord

b. Code Commit Guideline and Git Branching Strategy

There are two protected branches: **main** and **development**. They both require approval by another team member before any changes are merged. The main branch is where we perform the releases and deployment to production. The development branch is used to stabilize changes and integrate multiple personal feature branches. Everyone will create a feature branch from the development to work on and create a PR which targets development later. When merging to main or development branches we squash commits so we can have a single commit with a

description of the work done. This is particularly useful if we need to revert so we can just revert a commit.



Following the Git workflow as explained above a PR will need to be created and formatted with the following sections.

- **Summary**: Section about what the change and link any design or document.
- **Testing**: This section describes how the change was tested.
    I. **Manual**: Add information about manual tests that were performed.
    II. **Automated**: Add information about automated tests that were added.

c. Deployment Plan

Platform

- Heroku (Deploy with Git)

Necessary preparation

- Install Git
- Install Heroku CLI
- Create a Heroku Remote
    - Command for New App: heroku create -a example-app
    - Command for existing App: heroku git:remote -a example-app

Multiple Environments

- Create and link Development Environment
    - heroku create --remote staging //Step1
    - git push staging master //Step2
    - heroku ps --remote staging //Step3
- Create and link Production Environment
    - heroku create --remote production //Step1
    - git push production master //Step2
    - heroku ps --remote production //Step3

6. Quality Assurance Plan

a. Metrics

Metric Name	Description
Number of features	Number of features that were implemented; shows the complexity of the project.
Number of Defects	Number of defects reported in the project; shows the reliability and completion of the project.
Total Man Hour	The total time spent on or preparing for the project. This will allow us to track the total amount of effort spent in support of the project.
Test Passing Rate	Number of tests passed. This metric will be measured during each iteration, and the types of tests include unit tests and manual tests.
User Story Counts	Number of user stories created in Pivot Tracker. The number of completed user stories will be tracked in every iteration.

b. Coding Standard

Our team will be adhering to the Google Java coding standard found here. We will actively achieve this by configuring our IDE to implement the formatting of this coding standard (link).

c. Code Review Process

The team will work collaboratively on all the documents and review them in GSuite before they are exported and committed to the Git repository. GSuite comments will be used to open issues in the documents then comments will be marked done after issues have been resolved. When it comes to code, reviews will be done in GitHub via Pull Requests (PR) and **one** approval will be required before the changes are merged. If there are conflicts they will need to be resolved before any changes are merged to the main or development branch. Everyone on the team will be encouraged to **review all changes** to the repository and comments to open issues or ask clarifying questions. The reviewer will check the code and use the following review checklist.

1. Manageability
2. Architecture
3. Maintainability
4. Correctness

5. Invalid input/states
6. Usability
7. Reusability
8. Object-Oriented Analysis and Design (OOAD) Principles

After the PR is ready for review it should be posted in the group Pull Requests Thread in the group chat in Discord. It's everyone's responsibility to monitor the thread for review requests and promptly review the changes. The Git repository has the GitHub PR template committed so when a new PR is created it starts all the sections to be filled and the review checklist listed for the reviewer.

d. Testing

1. Unit testing: Unit testing will be conducted by the developer responsible for writing the code for each core method before submitting a pull request. The test methods and results can be recorded in the test report.
2. Manual testing: QA Leader performs manual testing of the core functionality after each iteration. The test methods and results can be recorded in the test report.

e. Defect Management

1) Defect Type:

- High Priority Defect:
  - Defects that halt the whole program from running, should be fixed as soon as possible.
- Median Priority Defect:
  - Defects that impact some of the functions in the program, should be fixed before iteration.
- Low Priority Defect
  - Defects that are hard to notice and do very little impact to the program, can be fixed anytime.

2) Defects Tracking:

The QA Leader will use Pivot Tracker to track fixed and unfixed defects and the fix rate at the end of each iteration.

3) Defect Resolution Process

i. Any team member who finds a defect will create a ticket in Pivot Tracker and assign it to the developer responsible for the feature with the following information.

- Severity level markers (High, Median, Low Priority)
- Description and screenshots of the defect
- Expected results

ii. Developers fix the defect and create a pull request with an "error" tag. The developer can have the teammate who found the defect specifically review this pull request and approve it.

7. References

The Ultimate Code Review Checklist
https://www.codegrip.tech/productivity/the-ultimate-code-review-checklist/?utm_source=website&utm_medium=blog&utm_campaign=best-practices-for-code-review-process

Java Coding Standard
https://google.github.io/styleguide/javaguide.html

Atlassian Git Workflow
https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Generate Project Name and Logo
https://namelix.com/

Agile Scrum Process
https://powerslides.com/powerpoint-business/project-management-templates/agile-scrum-process/

8. Glossary

- PR: Pull Request
- IDE: Integrated Development Environment
- CI: Continuous Integration
- CD: Continuous Development
- API: Application Programming Interface
- DB: Database