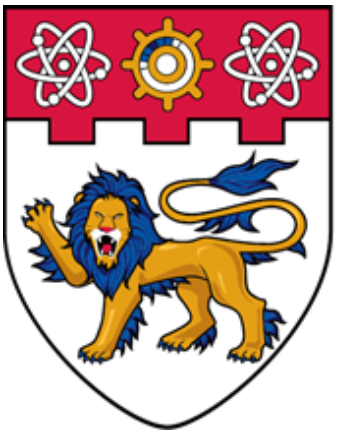

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ3002 Advanced Software Engineering Project:

Design Report on Software Maintainability - CashTrack

Version 1.2

Ravishankar Amrita (U1822377F)

Datta Anusha (U1822948G)

Kumar Mehul (U1822146E)

Alex Leong (U1921599D)

S Sri Kalki (U1921575L)

Nicklaus Tan (19403385F)

Elliott Ong (U1922981C)

Daniel Loe (U1921408A)

Lab Group: B2

School of Computer Science and Engineering (SCSE)

REVISION HISTORY

Version	Date	Description of Changes
1.0	23th March 2021	Initial Design Report - Software Maintainability draft
1.1	29th March 2021	Refined Architectural Design Patterns
1.2	31st March 2021	Finalised Changes

TABLE OF CONTENTS

1. Introduction

Software Maintainability refers to the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changing environment.

Over the span of its lifecycle, our CashTrack product is bound to undergo various software maintenance procedures such as fixing bugs, optimizing existing functionality, replacing unwanted features, fixing security vulnerabilities especially those found in the reused third party components.

In conclusion, maintainability is a critical consideration of any software system design, to ensure a quality product outcome. Hence, it is important to plan maintenance into the development lifecycle so that software may be maintained efficiently.

2. Design Strategies

2.1. The Planning Phase Before Development

During the process of initiating and designing CashTrack, the team performed analysis on the application scope, to predict the nature of improvements that would be required and implemented after the release of CashTrack.

Our vision for CashTrack involves a large number of users spread across a diverse range of population. Thereby implying a heavy traffic load on the application. Thus, scalability was one of the key factors that was kept in mind while designing the system architecture.

CashTrack is an application that focuses on seamless and exciting user experience. This motivated our team to adapt the Model-View-Control (MVC) model as our architectural framework, as MVC is best suited for a Graphical User Interface (GUI) application and allows for good modularity.

In order for CashTrack to remain fresh and useful in the market, our team decided that extensibility is another key factor we should keep in mind while designing the application. MVC allows for efficient and effortless extensibility, without significantly increasing the complexity of the program.

2.2. The Process of Developing

While creating new components of our application, our team made sure to follow SOLID principles. This allowed our application code to remain modular, facilitating maintainability, flexibility and extensibility.

Each component performs only a single function, and components such as confirmation pop-ups are reused in different components rather than writing redundant code multiple times.This keeps our code clean.

The technology stack that was utilised for CashTrack, further enables maintainability. For example, MongoDB provides a flexible database, allowing implementation of new features and saving new content into existing components and existing content respectively.

As an integrated part of our iterative product releases, test driven development is conducted through a series of small tests. Our core features such as creation of personal and shared expense records, shall undergo both alpha and beta testing. On the other hand, less significant features, such as user chats, shall undergo alpha testing prior to product launches. Due to safe distancing measures and contact minimization in place in response to COVID-19, beta testing has been put on hold as we were unable to get members of our target user base to test our product. As an alternative approach, team members, non-developers and developers alike, perform the role of testers and provide continuous feedback on the design and usability of the application.

2.3. Maintainability: Correction by Nature

Under this maintainability approach, corrective changes shall be made to our CashTrack application while it is undergoing testing. A detailed bi-monthly testing schedule shall be followed, to implement the following maintainability practices:

2.3.1. Corrective Maintainability

Features shall be tested thoroughly for fault detection, after which these faults shall be mitigated appropriately.

2.3.2 Preventive Maintainability

Upon no fault detection, features shall be tested thoroughly for robustness and potential faults or vulnerabilities, after which preventive changes shall be made as appropriate.

2.4. Maintainability: Enhancement by Nature

Under this maintainability approach, corrective changes shall be made to our CashTrack application concurrently as it is functioning in the market. A detailed bi-monthly testing schedule shall be followed, to implement the following maintainability practices:

2.4.1. Adaptive Maintainability

As technologies evolve and new operational environments emerge, features shall be tested for adaptability and maintainability, after which these features shall be updated for adaptability accordingly.

2.4.2. Perfective Maintainability

Under this approach, maintainability is implemented in a highly agile manner. After product delivery, the key aim remains to quickly detect an error and correct it, reducing maintenance costs and time required.

2.5. Maintainability Practices

To uphold quality in both process and product, the team has implemented the following maintainability practices over life cycle of our CashTrack product:

- Readable code
- Version Control
- Standardize Documentation
- Continuous integration makes the code easier to build and test
    - Automated build make the code easy to compile
    - Automated tests make it easy to validate changes
- System Modularity
- Design for maintainability from the outset

3. Software Quality Attributes

Some of the main Software Quality Attributes followed while planning, designing and developing CashTrack are as follows:

- **Reusability**

  Developers of CashTrack have ensured that the components like the Expense Cards and Summary Cards are made reusable to use in other components with some minor tweaks. This reduces the development and implementation time, and ensures that there is no duplication of components, in line with the DRY (Don't Repeat Yourself) coding principle.

- **Maintainability**

  CashTrack is built to undergo changes with a degree of ease. It is built with high cohesion and low coupling and thus changes can be to specific components, services, features as well as interfaces without causing any malfunction. Proper documentation and manuals are developed to reduce the onboarding time of a new developer wanting to make a change to the application.

- **Conceptual Integrity**

  General good coding and design practices like following the SOLID principles, using helper methods and doing pair programming sessions were followed while developing CashTrack to maintain consistency and coherence. All the components follow the same coding style even though it was developed by different developers, and future developers are also encouraged to follow the same pattern in the developers' guide.

4. Architectural Design Patterns

CashTrack uses a MVC (Model-View-Controller) architecture design for fast development, low coupling, high coupling and easy maintenance.

Models represent the shape of the data and help to retrieve and store the data in a defined format. This is stored as records in the MongoDB database where the shape and parameters of each record has been pre-defined.

Views are the pages that the user can see on the browser screen when using CashTrack. All of the views can be found in our source code in the ‘front-end’ folder, and we used Angular to develop them.

Controllers contain the business logic for the updation of the views whenever the user interacts with it, by saving and getting data from the database. We’re using REST APIs for the views to access the particular controllers to provide modularity and easy tracking of errors. All of the controllers can be found in our source code in the ‘back-end’ folder, and we used ExpressJS and Node to develop them.

Using MVC allows us to have multiple views which is essential for CashTrack as it follows a ‘dashboard’ structure. All of the views can be updated independently using their respective controllers, and data integrity is always maintained in a decoupled database. MVC also supports asynchronous updates which is important for features like Notifications and live updating of charts on the Home Page (Dashboard).

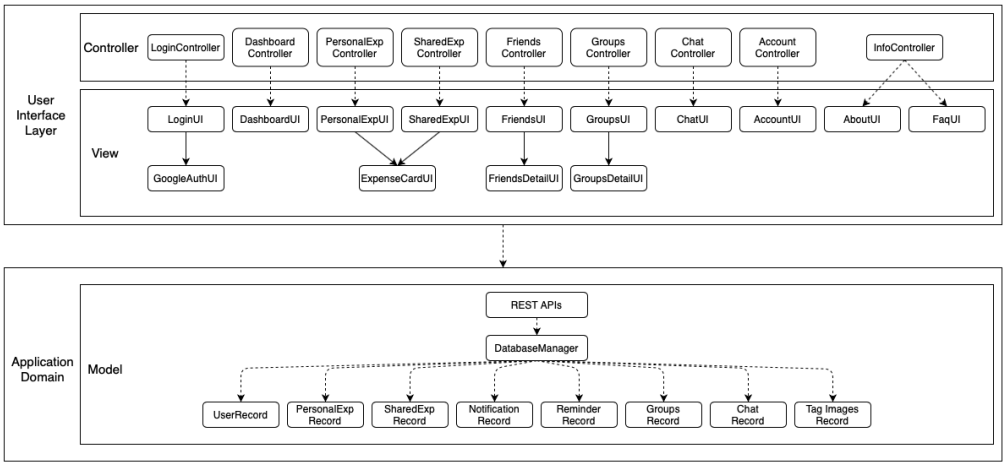

Figure 1. Architectural Design Pattern

4.1 Design Patterns

4.1.1 Client and Server Pattern

CashTrack has adopted a client-server design pattern by using ExpressJS with Node to create REST APIs, and the Angular front-end following a request-response for the updation of views on user interaction.

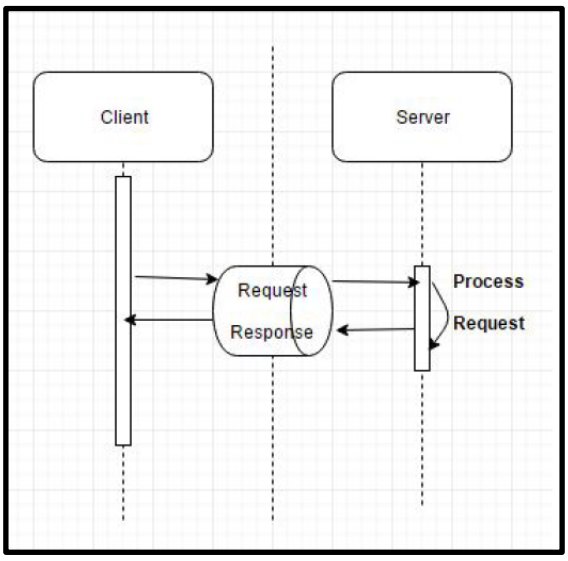

Figure 2. Client and Server Pattern

Whenever an Angular View needs to be updated, it makes a Request to the Express Server via the REST APIs through HTTP Protocol. The Express Server processes the request based on the endpoint, gets data from the database, and provides an appropriate Response back to the View which made the request.

As all of the requests are asynchronous, the response from the server is fast and in a defined format which makes it easier for the client to process it.

4.1.2 Observer Pattern

Observer Design Patterns allows for some components, called the subjects, to maintain a list of dependencies and will be notified if there is a change in any of them.

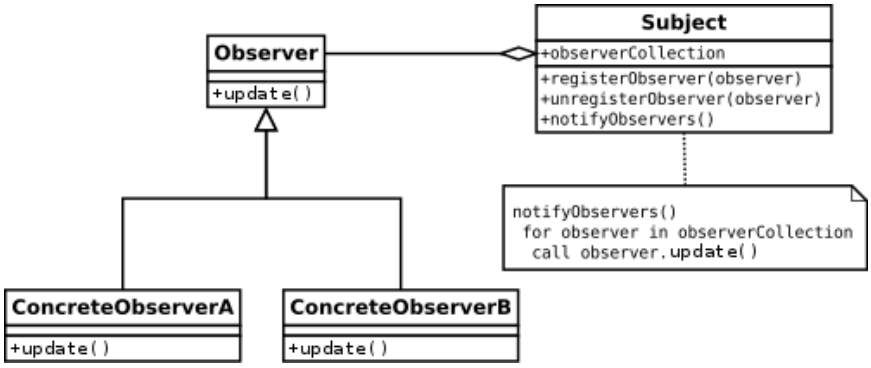

Figure 3. Observer Pattern

This design pattern is implemented in CashTrack in several places, like the Notification and Dashboard Components. Whenever there is a new notification generated for a particular user in the database, the NotificationUI component gets updated for the user to interact with it. Similarly, if a new personal or shared expense is added for a user, the Dashboard Component is updated to reflect the new Total Spending, Money In/Out and the pie and line charts are updated to show the new changes.

Using this design pattern promotes maintainability by decoupling different components that have to interact with each other.

5. Software Configuration Management Tools

This is where we will discuss version control management, and tracking on who made what changes and when.

5.1. MediaWiki

MediaWiki is an open source collaboration and documentation platform. It provides comprehensive documentation through its functionality and features. It also provides a wide range of capabilities to enable users to document their information using various styles. Further, it allows users to concurrently edit information thus making an efficient use of time and preventing loss of information during parallel edits.

5.2. GitHub

GitHub is a source code hosting platform using the distributed version control and source code management Git. One of the most important features that GitHub provides is bug tracking. It allows users to create an issue ticket, to describe an issue regarding a software bug or documentation, and allows the user to assign it to other team members. All collaborators receive updates on the progress of the project and the issues.

5.3. OneDrive

Runtime Terror utilised OneDrive for file storage and as a document reserve for documents created originally. OneDrive enables the team to share and store files amongst team members. It also provides editing capability along with document version control. Furthermore, any document on OneDrive allows users to utilise the full functionality of Microsoft Office applications.