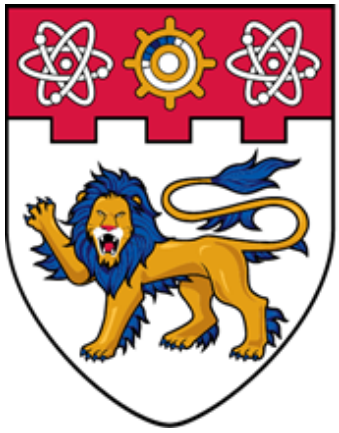


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

# CZ3002 Advanced Software Engineering Project:

# Design Report on Software Maintainability

# Kim's Convenience Modern Cinema

*Version 1*

**School of Computer Science and Engineering (SCSE)**

# REVISION HISTORY

| Version | Description of Changes | Approved by | Date |
|---|---|---|---|
| 1.0 | Initial Design Report - Software Maintainability draft | | 11 Oct 2021 |
| 1.1 | Refined _____ | | |
| 1.2 | Final Version | | |

# TABLE OF CONTENTS

# 1. Design Strategies

## 1.1. The Planning Phase Before Development

While initiating and designing Kim's Convenience Modern Cinema (KCMC), we have analysed the application scope to predict the possible improvement to be implemented in the future after the release.

1. Scalability
   In the near future, we envision KCMC to be widely used among Singapore's major cinema chains which have millions of patrons. This implies a heavy and high traffic rate. Therefore, scalability is one of the targeted key factors to be kept in mind while designing the system architecture.

2. Extensibility
   Additionally, we aim for KCMC to stay relevant and up to date to the market, hence extensibility another important factor. This would allow for efficient and effortless extensibility, without significantly increasing the complexity of the program.

Since KCMC places great importance on the user experience and a seamless service, we have adapted 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. This allows for ease for change without affecting other functionalities.

## 1.2. The Process of Developing

The SOLID principle was closely followed:

- Single responsibility
- Open–closed
- Liskov substitution
- Interface segregation
- Dependency inversion

This has supported our application code to be more understandable,

flexible, and maintainable.

As an integrated part of our iterative product releases, test driven development is conducted through a series of small tests.

Due to safe distancing measures and contact minimization in place in response to COVID-19, team members, non-developers and developers alike, perform the role of testers and provide continuous feedback on the design and usability of the application.

### 1.3. Maintainability: Correction by Nature

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

#### 1.3.1. Corrective Maintainability

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

#### 1.3.2 Preventive Maintainability

If no fault gets detected, features will be tested thoroughly for robustness and potential faults or vulnerabilities, after which preventive changes shall be made as appropriate.

### 1.4. Maintainability: Enhancement by Nature

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

#### 1.4.1. 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.

#### 1.4.2. Perfective Maintainability

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.

### 1.5. Maintainability Practices

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

- system modularity
- version control
- configuration management
- consistent integration effort
- couple architecture components loosely
- automate development pipeline and tests
- write clean code

# 2. Architectural Design Patterns

## 2.1 Model-View-Controller Pattern

KCMC uses a MVC (Model-View-Controller) architecture design. MVC allows for

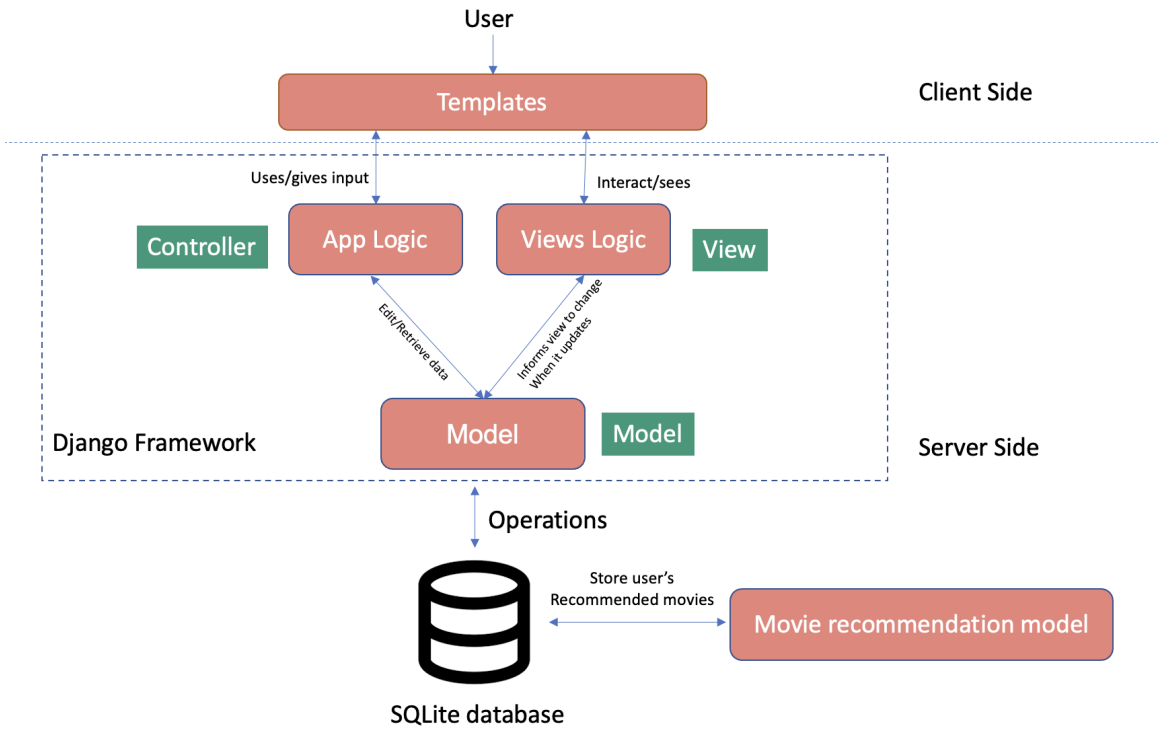
- fast development;
- low coupling;
- high cohesion and;
- easy maintenance

MVC allows for multiple presentations of an object and separate interactions with these presentations.

**Model** holds data and operations for achieving computation goals of KCMC. It helps to retrieve, store, update and delete the data in a defined format. Django provides support to create these models with a database. The records are stored in the Django database with pre-defined parameters for each record; it can be found in the “db.sqlite3” and “models.py” files.

**View** is the interface(s) that is displayed on the browser screen when users are on the KCMC webpage. It also controls what the cinema goers will see and interact with when on the application. HTML and CSS were used to develop these interfaces in the form of view templates. Django integrates these view templates to implement the front end of the application. All of the views can be found in our source code in the ‘templates’ folder.

**Controller** receives and carries out commands from the users through the views. It contains the business logic for interaction between the user inputs and the data in the database. The application logic is mainly written in Python and JavaScript. The controllers can be found in javascript files located at ‘static/js’ and various python files such as admin.py or views.py.




## 2.2 Client-Server Pattern

KCMC also uses the client-server pattern. This pattern is a distributed application structure which separates tasks between the service providers called servers and the service requesters called clients.
This pattern allows for:

- greater security due to its centralised architecture where all data is stored at a single controlled location;
- scalability as more servers can be easily connected to allow for more concurrent users;
- accessibility since every client is provided with the opportunity to access the system irregardless of their location.

**Client** here refers to the machine a user uses to access the KCMC web application. When they first access it, all view templates will be first downloaded and stored in the user's browser cache for quicker access in the future. The user's viewing preferences are also stored in the browser cookies throughout their entire time on the application to improve their user experience. The browser uses these templates and cookies and acts as a generalised user interface device. The browser will allow users to interact with the system such as requesting for information. The browser will send these requests to the KCMC server through HTTP protocols  As all of these requests are asynchronous, the response from the server will be fast.

**Server** here refers to KCMC servers which fulfill the service request from the multiple client components. It is also the principal access point for all client browsers. The server holds all data and information pertaining to the application and the application logic. Depending on client request, the server may also initiate certain server-side processing on the data or view of the browser. KCMC application uses client-server rendering. This means no new request is made to the server whenever the user enters a new page. Content instead is rendered on the client side browser using the JavaScript library.

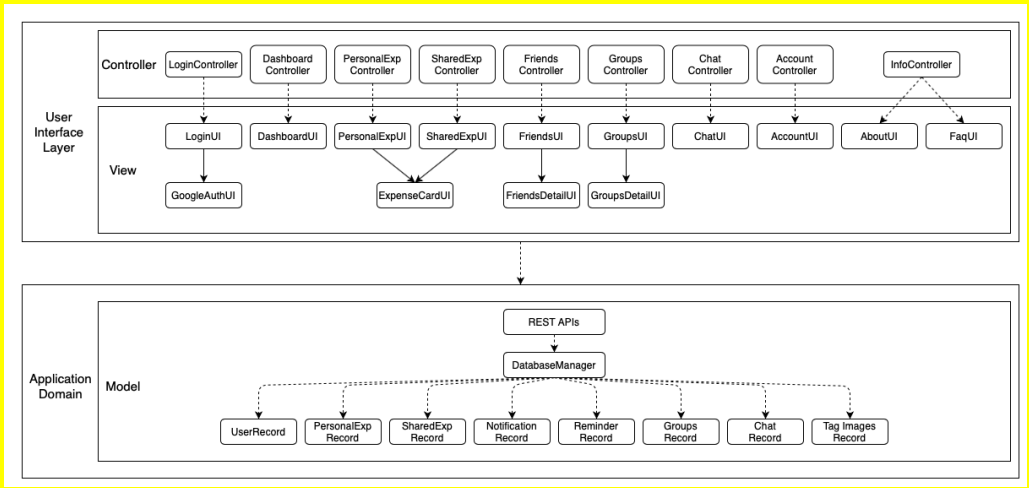



*Figure 1. Architectural Design Pattern*

## ~~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.~~

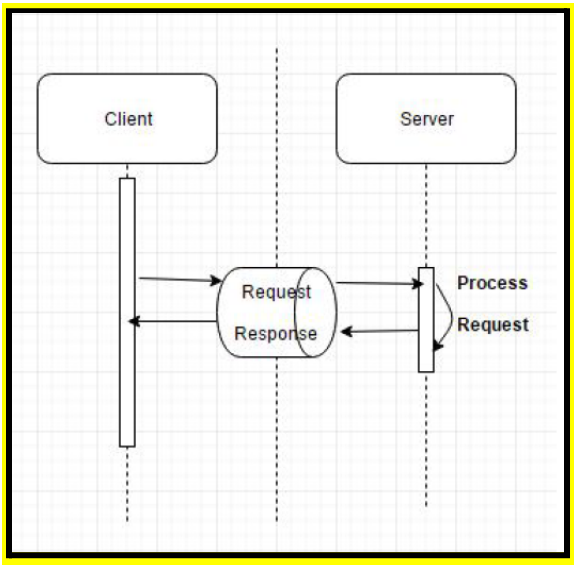


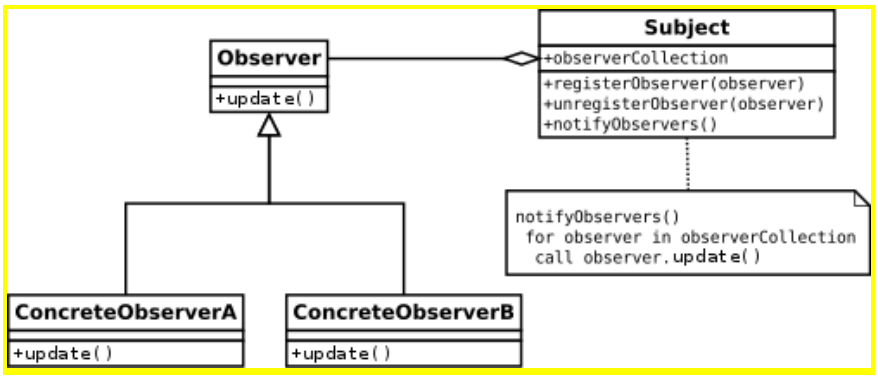

*Figure 2. Client and Server Pattern*

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.~~

## 2. Software Configuration Management Tools

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

### 3.1. MediaWiki

MediaWiki is a free and open-source wiki software which supports collaborative documentation. The software is optimized to efficiently handle large projects, which can have terabytes of content and hundreds of thousands of views per second. Additionally, its wide range of functionality and features have allowed for comprehensive documentation. One key feature is its ability to allow users to concurrently edit information, in return preventing any loss of information and also allows efficient use of time and manpower.

### 3.2. GitHub

GitHub is a source code hosting platform using the distributed version control and source code management (SCM) Git. It provides access control and several collaboration features such as bug tracking, feature requests, task management, continuous integration and wikis for every project. 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.

### 3.3. Google Drive

Google Drive is a file storage and synchronization service developed by Google. It allows users to store files in the cloud, synchronize files across devices, and share files. This allows for users to edit information at the same time, thus making an efficient use of time and preventing loss of information during parallel edits. Having everyone up to date to the latest documentation also prevents miscommunication.