|  |  |
| --- | --- |
| Analysis Report | Course  Design Patterns  Course Instructor  Aisha Urooj  Group Members  M. Fawad Jawaid Malik K112116  Shoaib Ahmed K112016  Zohaib Masood K112114  Syed Muhammad Zohair K112181 |

# Introduction

“Virtual Receptionist” is an Android Application which helps organizations and their employees to manage meetings and appointments as well as helps them to easily exchange documents among other employees of the organization. It also helps people, who are not part of the organization, by allowing them to make appointments with the employees of the organization. This application can be customized according to the theme, color codes and logos of the organization once it is bought.

Employees can use this application to call meetings with other employees. The employees called in the meeting will also get notified. Employees can also view their scheduled meetings as well as call off or postpone meetings they called. The visitors who make appointment with any employee using this application needs an approval from the employee, for this purpose the application provides an interface to view approved, pending and missed appointments. Another important usage of this application is to sending and receiving of documents among employees. This is usually done in organizations if an employee needs approval or review on documents from other employees.

The visitors using this application can make appointments for interview, surveys or any other purpose. They are notified every time an appointment changes its status, for example, an appointment is confirmed or postponed by the employee.

# **Architecture**

“Virtual receptionist” application has been made for the office communication purpose, which is virtually working as a substitute of a receptionist to the office system and helping all its client to communicate, making appointment etc. So its **architecture** splits into 5 layers i.e. User Interface (UI) Layer, Android API Layer, Business Logic Layer, Network Layer, Database Layer.

## UI Layer

This is the Layer which is working as a Graphical user interface of the Virtual receptionist system. This layer contains all the buttons, navigation components, UI components. This layer connects with Android API Layer to provide navigation, event dispatching and handling and exception handling in the application. The application focuses on a simpler UI because, the simpler a UI Layer is, the easier a user will find it to use.

## Business Logic Layer

Business logic layer is core layer in the Virtual Receptionist System which calculates and send information to users. This layer contains different algorithms which calculates different activities like reminder of meetings, appointments, sending of files etc. This layer uses services from Android APIs.

## Android API Layer

This layer represents the built-in APIs of Android. These APIs are used to handle different components of the application like Dialog box handling, database and network handling etc. This layer mostly covers the tasks which are delegated to the core of Android application.

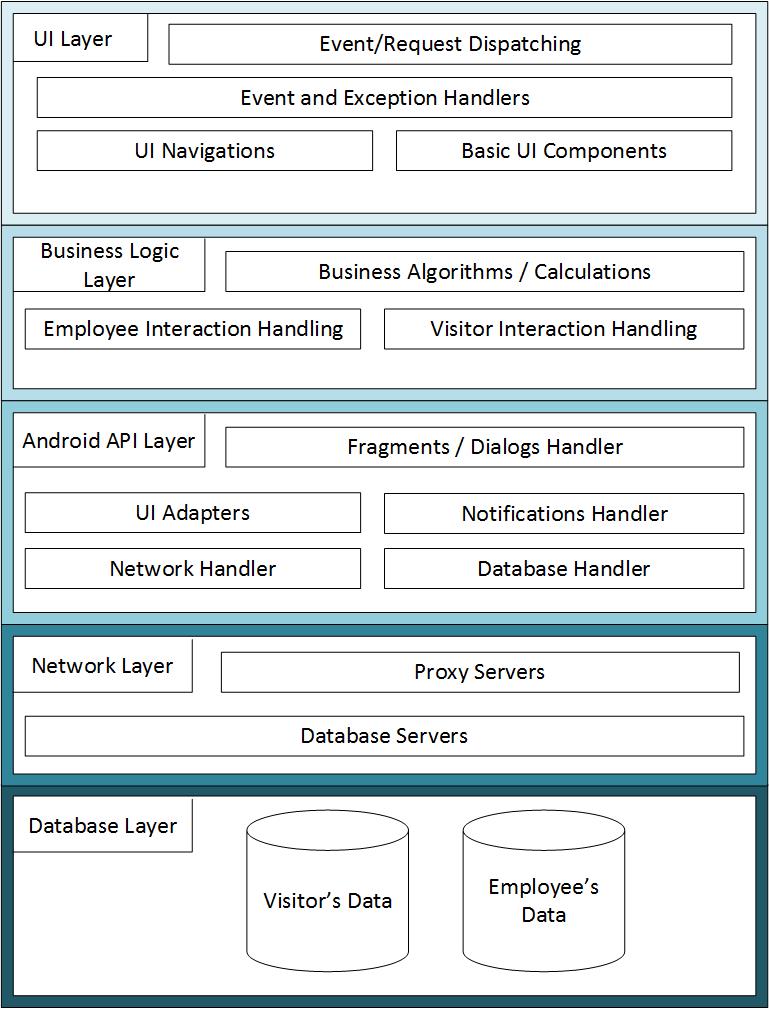
## Network Layer

Through this layer the data will be transferred, by the user and to the user, which is stored in database system. The user is connected to the database system through this layer. Network layer also helps to maintain, manage and update data of the database system through queries by user. Another task of this layer is to control access to remote database by users over the network. It uses proxy servers to control access of less or non-privileged users to critical database.

## Database Layer

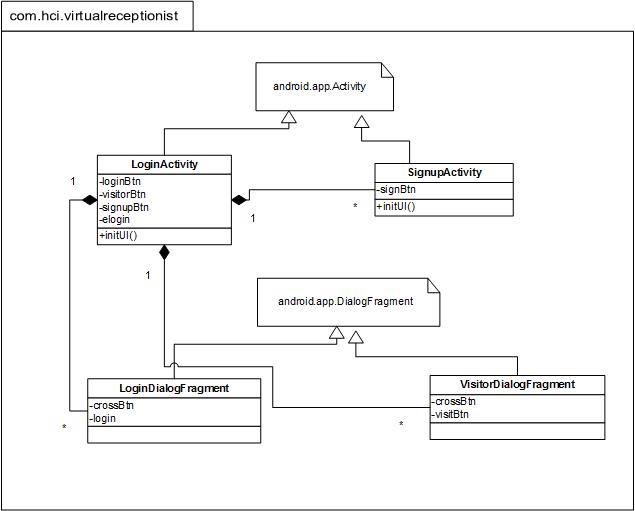
In this layer it will store the data of the users i.e. their personal data like name, email, address, phone number, schedule dates, pictures etc. It is the most important and primary layer of this application architecture on which whole system will rely on, through which every user will connected and get updates when there is any.

# **Architectural Diagram**

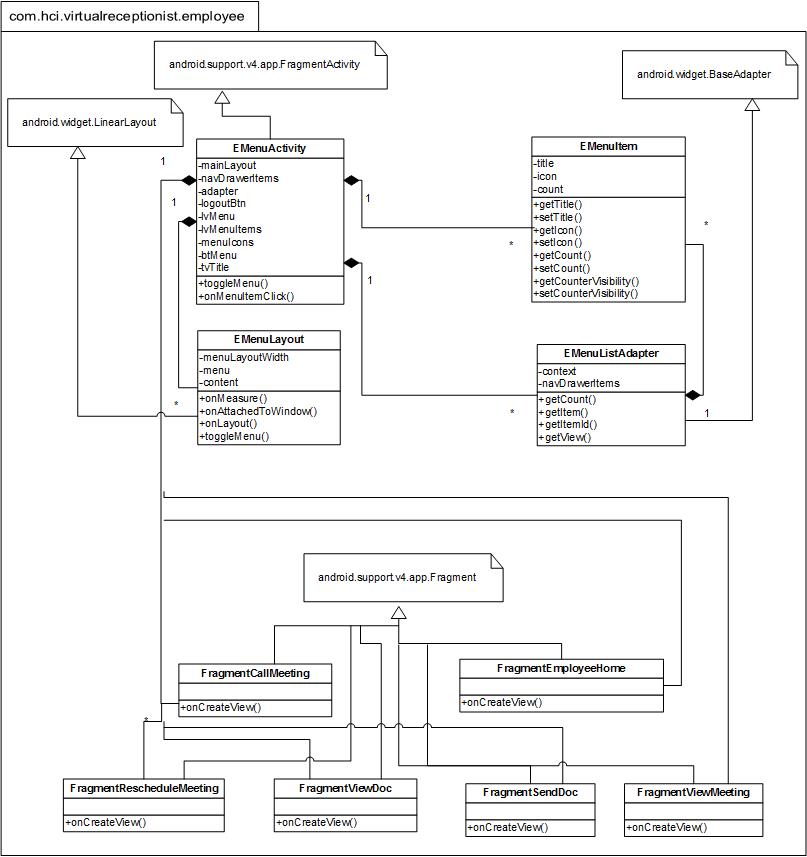


# **Class Diagrams**

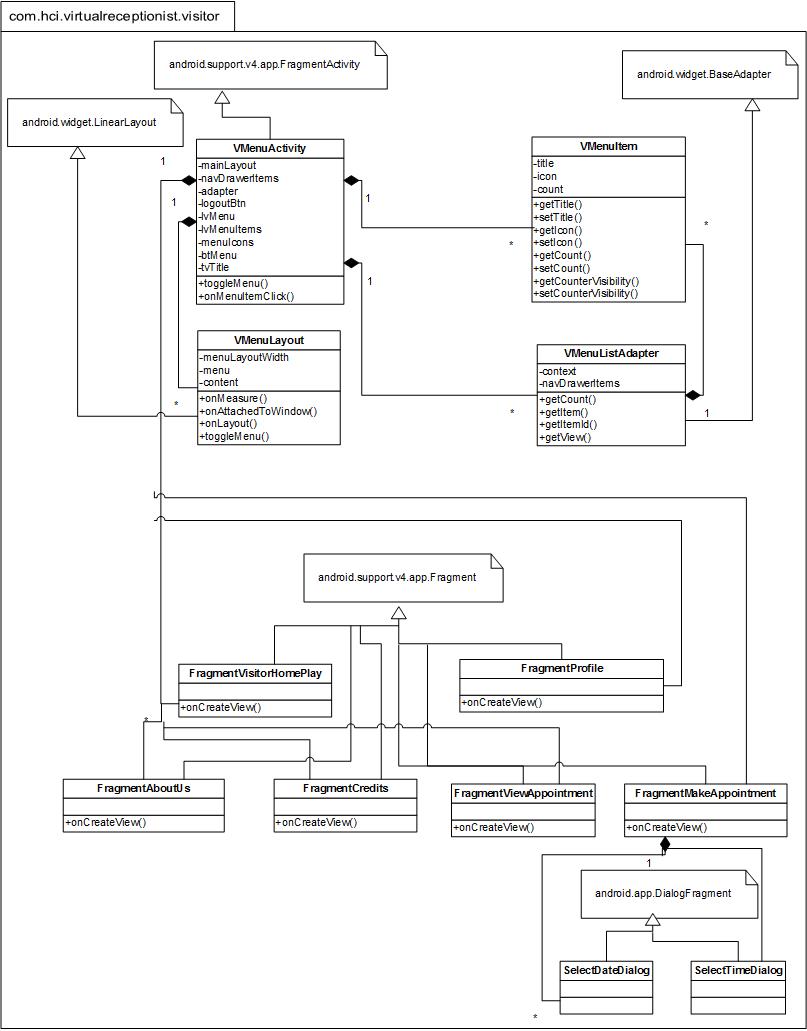
## Main Package



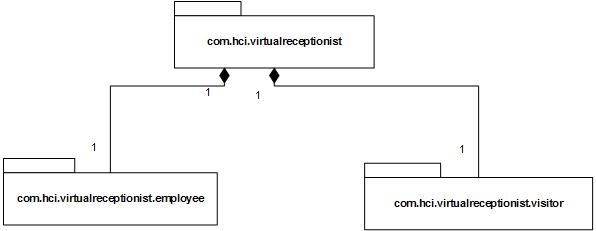
## Employee Package



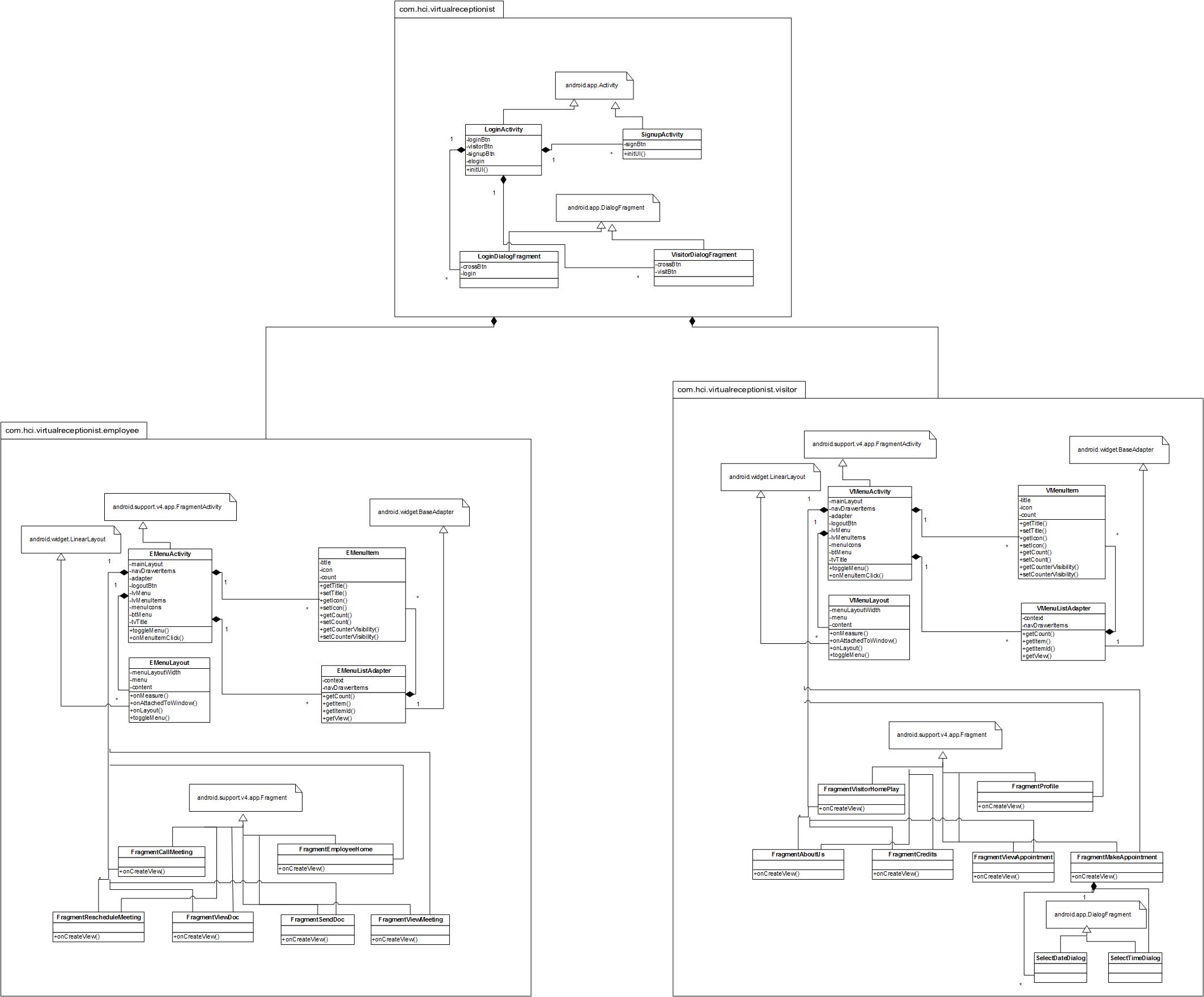
## Visitor Package



## All Packages



## Complete Class Diagram



# **Improvements**

The improvements which can be made in current code of Virtual Receptionist are as follows:

1. There are multiple classes which are duplicated due to type of login. The behavior application adopts after login for a visitor and employee can be set at runtime by using the “Strategy Pattern”. We can make a single login interface and decide on runtime whether use the employee behavior or employee behavior depending on the login ID and Password. I.e. if it is left blank, the user is a visitor. This will decrease the number of duplicated classes as well as reuse classes.
2. The user of the application, whether employee or visitor needs to be notified about the appointments or meetings being changed, so we can use the “Observer Pattern” to improve this behavior.
3. “Singleton Pattern” can also be used in the current code, as the employee object is quite expensive than visitor object, so it should be made only when needed and should be reused throughout the application runs. This will also help to provide a global point of access throughout the code.
4. There a number of collections throughout the code. For example, pending meetings/appointments, confirmed meetings/appointments, passed meetings/appointments pending documents and reviewed documents. These collections have different data structures, for example, meetings/appointments are hash tables as they are filtered by the person who called the meeting or who made the appointment. This can be done by using “Iterator Pattern” as it would help to cater all the collections using a single interface.
5. Our application uses remote objects to be accessed time to time and it also make a vast use of network through which employees and visitors are connected. In our application, one of our users is the Head of Organization. He is concealed as an employee to control his access. We can use “Remote Proxy Pattern” to cater this scenario in our application. We can also use “Virtual Proxy Pattern” in our application while the documents are loading, when an employee needs to access it.
6. The “Adapter Pattern” can be used to adapt how a list of documents or meetings is showed. We can make an adapter which adapts the data from meetings or documents and make different data view in the same way.
7. The “Model-View-Controller” pattern can also be applied to improve our application as it will collectively include most of the patterns or improvements discussed above.