Sports Center Management Software
Architecture Notebook

1. Purpose

Purpose of Architecture Notebook is collecting all properties about system design and standards in one file. These properties include architecture goals, philosophy, assumptions and dependencies, the requirement of design, architectural mechanism, design constraints, justifications, and decisions. At this point, we can easily check whether we follow the rules or not the project after each step. Besides, we can have a standard on the Architectural side of the project. So, we can easily say this document concretize the properties of the system and it helps to the new team member for adapting to the project.

2. Architectural goals and philosophy

In this project, we aim to design sports center's mobile and web application. At this point, our project is divided into two different sides. These are member and manager sides. In member side, we can have a wide variety of user profiles. So, the first architectural goal is simplicity. It is important to us that users can use the application easily. The members may want to connect with many different devices. Therefore, we need to provide various device support. (For this title, we decide to use Android version 4 or newer). In the mobile application, we aimed to keep the size of the mobile application and system requirements as less as possible. So, we can reach maximum population of members. In management side, we need to share statistics of centers clearly with managers. A misunderstanding at this topic can cause bad sequences for the firm. Our system doesn't need any additional hardware specializations. In website, we work on modern web browser. But, we plan to increase browser support.

3. Assumptions and dependencies

In this project, we will develop web and android application for sports center management. We use Angular5 for web application side because of Javascript languages are easier than other languages which are using for web application's implementation. Also, our team members have no knowledge about web designing and implementing so, we have to learn fast and pass on implementing.

We use and android because of implementing an android application's implementation is easier than other mobile platform applications, android is widely used and our team members have knowledge about it.

We use MySQL for database side, because of easier and simpler than other database services and our team have knowledge about it. Also, our server supports MySQL and server processes about database became easier.

We use java for control database, web and android applications' base. Firstly, we have to use and object-oriented language like C# or java. Secondly, we are more experienced about java programming. Also, java community is better than C# community, any problem's solution is easier to find for java.

4. Architecturally significant requirements

- The system should stay available at all time.
- Multiple users can access to system at the same time.
- Mobile device's version should be higher than 4.2.
- Application should have short response time for any processes.
- User should have a up-to-date web browser for web application.
- Internet connection is necessary for both applications.

4.1.1 Why Use the Spring Framework?

http://www.wrox.com/WileyCDA/Section/Why-Use-the-Spring-Framework-.id-130098.html
http://www.onlinetutorialspoint.com/spring/advantages-of-spring-framework.html

4.1.2 Why JavaScript Is and Will Continue to Be the First Choice of Programmers?

https://dzone.com/articles/why-javascript-and-will
https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/
https://rubyroidlabs.com/blog/2016/10/angular-2/

4.1.3 Why is the Android OS So Popular?

http://opensourceforu.com/2016/02/why-is-the-android-os-so-popular/
https://medium.com/@developer45/5-advantages-of-android-app-development-f06c5a9283d1

4.1.4 Why MySQL is still king?

https://www.infoworld.com/article/3195764/nosql/nosql-no-problem-why-mysql-is-still-king.html
https://www.smartfile.com/blog/the-pros-and-cons-of-mysql/

5. Decisions, constraints, and justifications

- The system should be developed in accordance with Object Oriented Design. OO design is important for future development and it is more useful for teams.
- The system's official language is Java. Besides that, we should use Angular 2+ in frontend side. In mobile application, we should use Java for Android. Java has great support in server side and it required for Android mobile application development. Angular 2+ is also use class structure, so it is most available front-end framework for us.
- Model-View- Controller is the system's architectural pattern. In this way, we can separate critical system parts easily. It decreases cost of maintenance.
- The system's database management system is MySQL. Because it is open source and we can easily learn how to use it.
- In mobile side, the application supports 4.2 or newer versions of Android. It provides wide support for maximum users.
- In website, the website supports Google Chrome, Firefox, Opera and Microsoft Edge. It provides wide support for maximum users.
- The Server is available for every time. So, we need to use cloud servers. In this case, we rent a server from Digital Ocean.
- In front end development, we use bootstrap 4 for better and modern view.
- The development environments; Visual Studio Code for Web development, Android Studio for Mobile application, and Sts tool for back-end development. It is important for team works. Because it decreases inconsistency.
- Web application runs on apache server. Its open source software and it has large support.

6. Architectural Mechanisms

Availability

The system will be active whenever the user wishes to access it on mobile and web application.

Archiving

The user's past training information will be stored in the database.

Graphics

User interfaces will be as simple as possible for the user's easy use.

Communication

The user will be able to make calls at any time using the contact information.

Event Management

System will show events to the user according to user situation.

Security

The user information will be kept encrypted in the database.

Data Management

All data related to the application will be kept using the MySQL database. Mobile and web application will use the data in the database via web service.

Memory Management

Some personal data will be kept in local memory on the user's device.

Mail

The user will be notified via e-mail when necessary.

7. Key abstractions

Users:

- Visitor: Whoever created an account but has not paid money,
- Member: Whoever created an account and paid money
- Trainer: Whoever is responsible for coaching members and planning their activity.
- Manager: Whoever is responsible for the management of the sports center
- Owner: Whoever owns the gym

Interfaces: It creates connection between users and system.

Models: It consists of system's main components.

- Courses
- Centers
- All user types

Database: It holds all information about the sports center and user

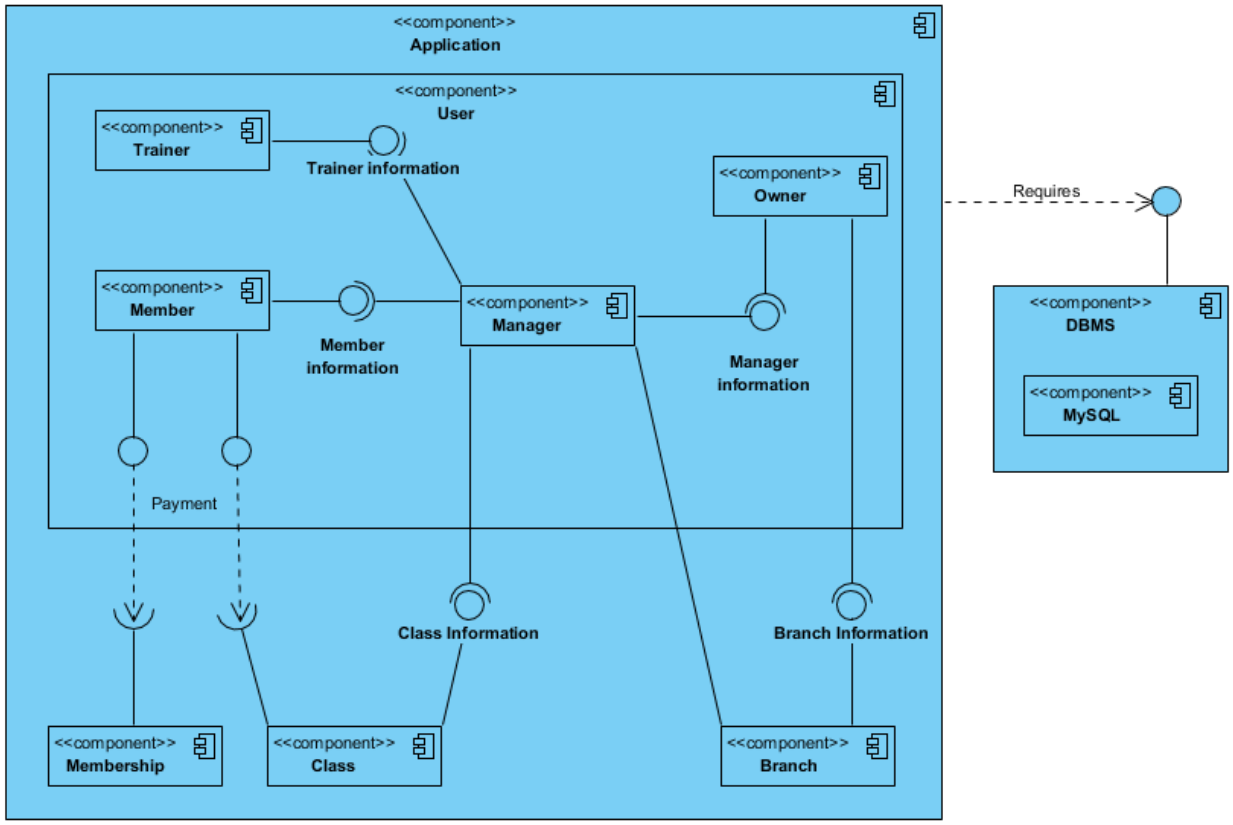

Figure 1: Component Diagram

8. Layers or architectural framework

The system consists of 3 different layers. The first layer is Presentation layer. This layer is responsible from the view of the project. We develop both mobile and web application. So, we have two different interfaces. These interfaces include many view classes. The second layer is the business layer. This layer is responsible from creating communication between the presentation layer and data layer. The business layer gets user's demand and provides them. The last layer is data layer. The data layer provides resources to the system. This layer includes database connection. The business layer reaches necessary information with the data layer. This layer is also responsible from the connection between other systems.

9. Architectural views

By using 4+1 architectural view model, we got logical, development, process and physical views.

Recommended views

- **Logical:** Describes the structure and behavior of architecturally significant portions of the system. This might include the package structure, critical interfaces, important classes and subsystems, and the relationships between these elements. It also includes physical and logical views of persistent data, if persistence will be built into the system. This is a documented subset of the design.

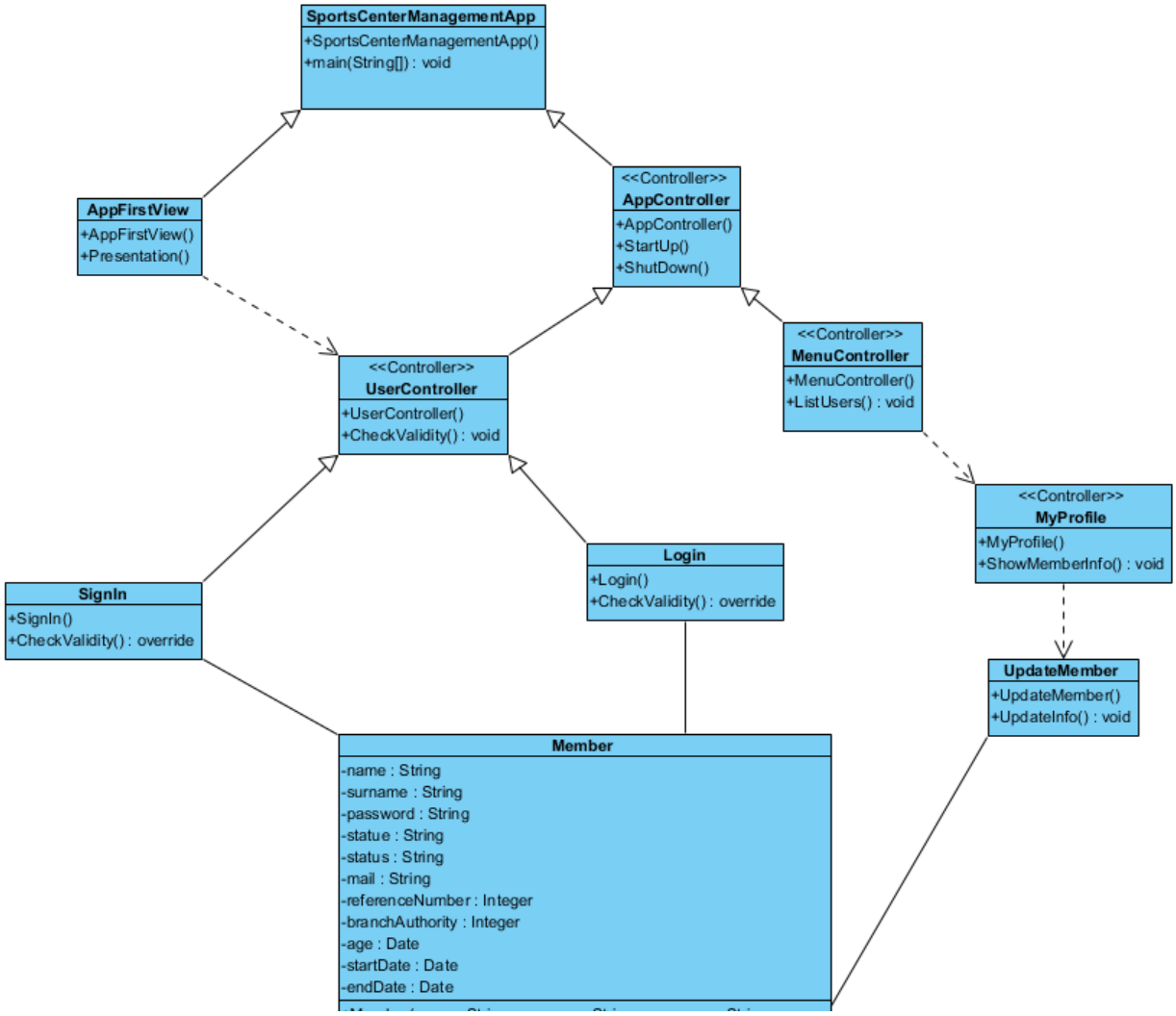

Figure 2.1: Class Diagram

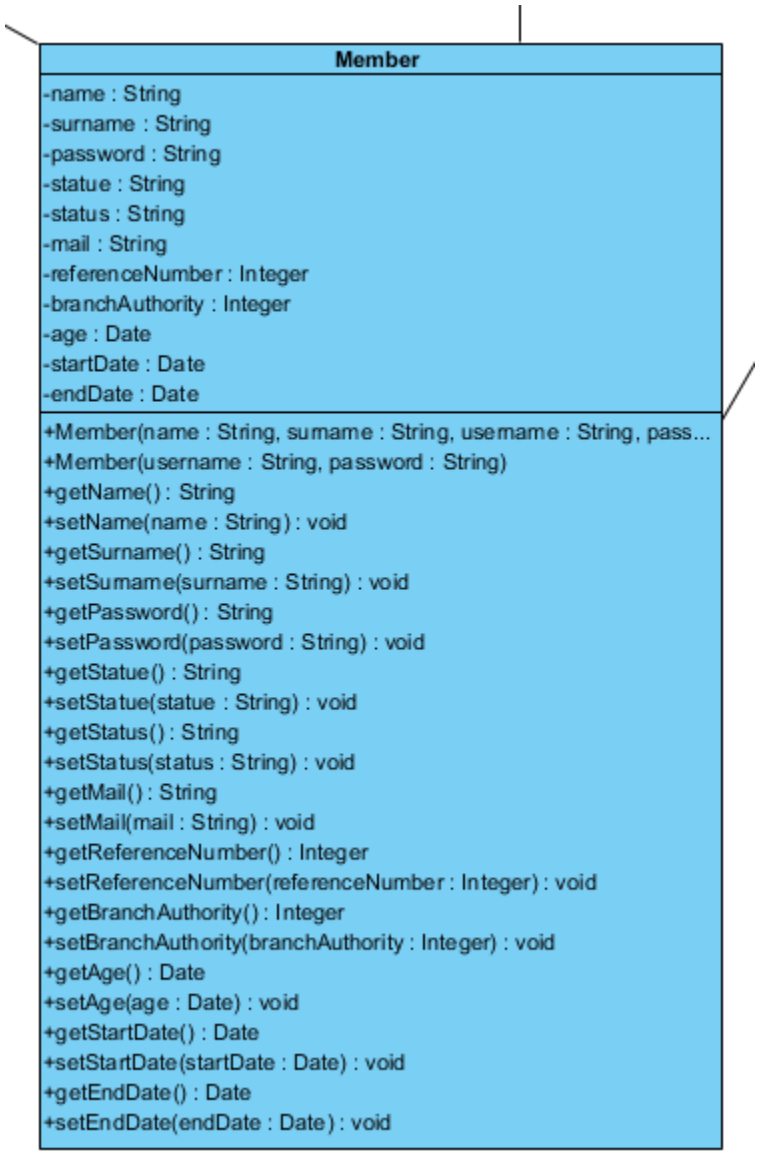

Figure 2.2: Class Diagram

- **Process View:** Describes the physical nodes of the system and the processes, threads, and components that run on those physical nodes. This view isn't necessary if the system runs in a single process and thread.

- **Development View:**

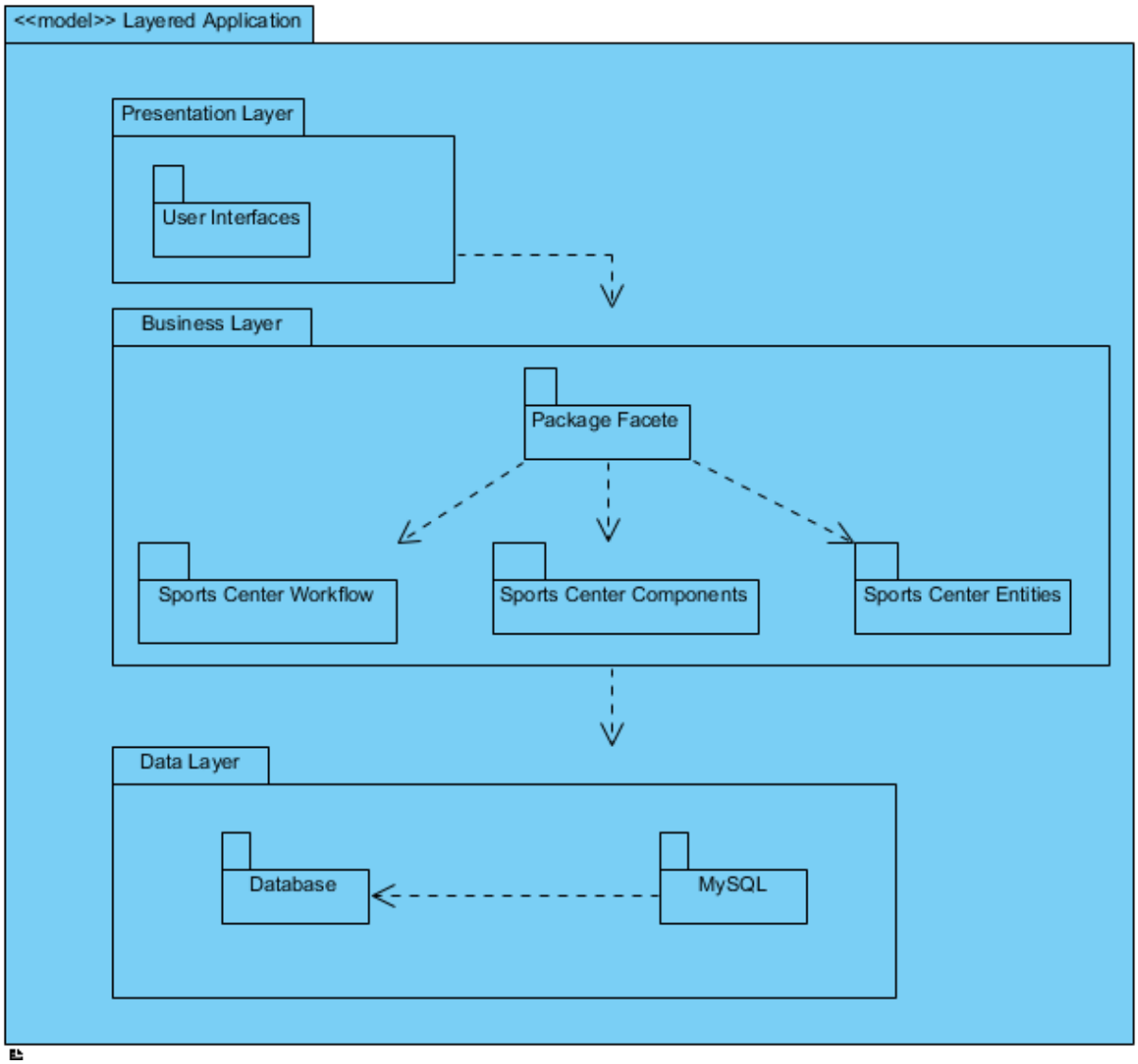

Figure 3: Package Diagram

- **Physical View:**

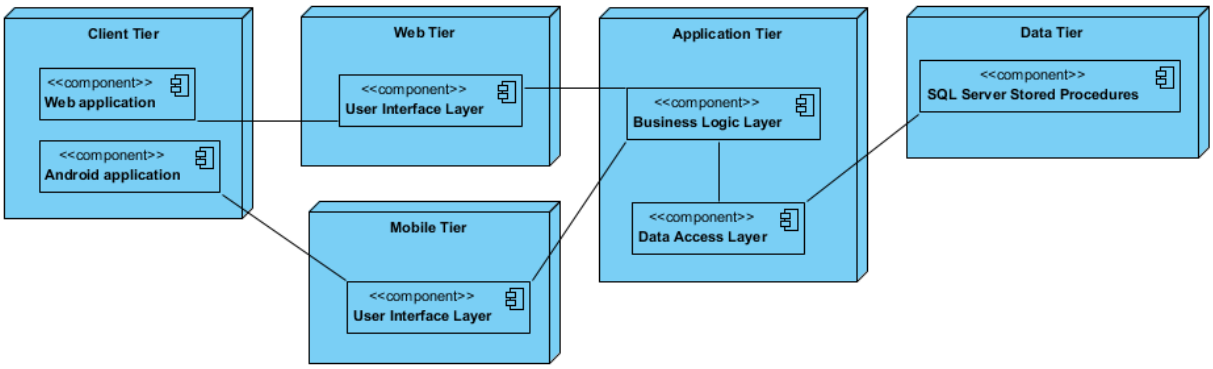

Figure 4: Deployment Diagram

Use case: A list or diagram of the use cases that contain architecturally significant requirements. If you want to look at the details of this diagram, you can check SRS Document's Appendix D.

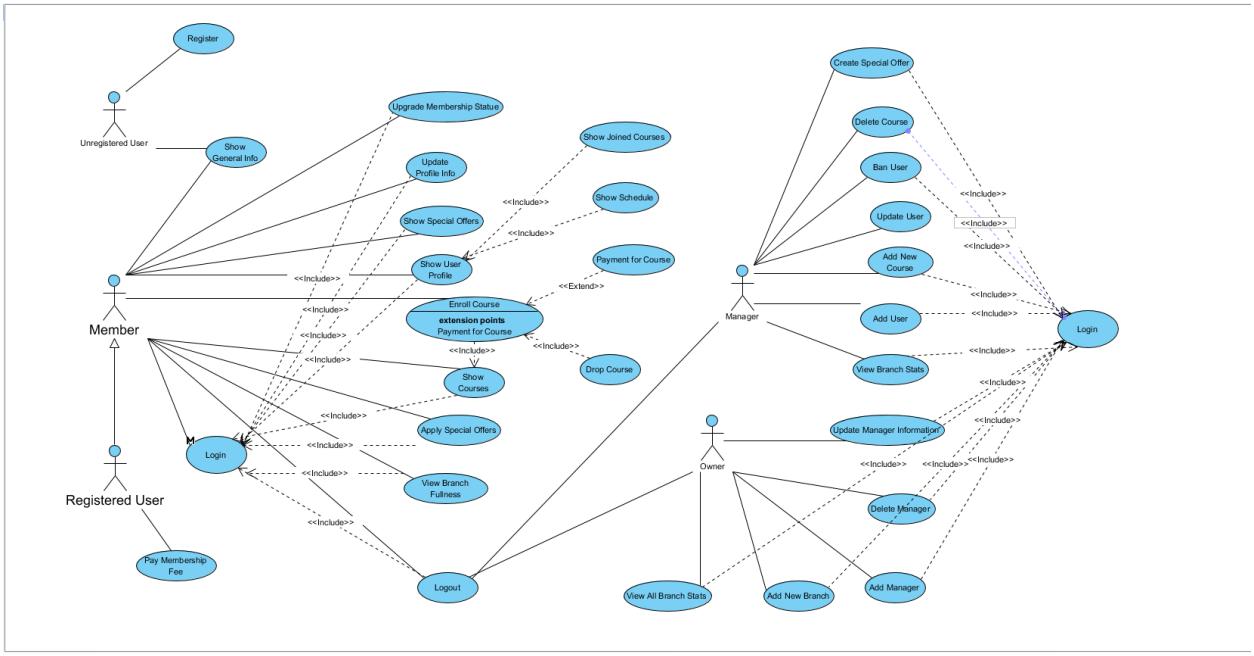

Figure 5: Use Case Diagram