Assignment 1 – ECE 452

Software Architecture Description Document

Atulan Zaman *a3zaman*
Jun Ho Lee *jh33lee*
Rashna Razdan *rrazdan*
Nadeem Ahmad *n9ahmad*

Abstract

This is the Software Architecture Description document for the Healthy Eating web application. The Healthy Eating application is a dynamic web application that users can utilize to keep track of the amount of calorie that they are consuming. This task is made more interesting by the fact that this application interacts with Facebook to link the user with all of their friends who also use the application for keeping track of calories. Furthermore, users can compete with their friends in order to determine who is superior in term of eating healthy.

The first section describes the architectural style and presents a system architecture context for this application. The rationale behind making these choices, followed by the design constraints and goals that are met by choosing the architectural design are also mentioned. The major use cases of the application are described using Use Case diagrams, followed by Sequential diagrams to illustrate certain key use cases. The viewpoint section describes some of the design concerns that were relevant to the project but which were not mentioned as part of the architectural features in the previous sections. Finally the system level deployment of the application is presented for a higher level view of the hardware involved.

Table of Contents

Table of Figures

1 Introduction

1.1 Scope

The scope of this document is limited to the describing the software architecture for a client level understanding of the application. The information presented is based on the development team's knowledge before the software implementation of the application. This document does not contain details regarding the specific class diagrams and their relationships since this information is more relevant at the development phase of the application. It also does not provide any details about the API support for the program because this is also mainly relevant in the development phase of the application.

1.2 Purpose

The purpose of this document is to describe and justify the architectural decisions and design choices made during the planning of the application. The document also offers insight into the constraints made during this process. Additionally, this paper describes the key features of the application in terms of Use Case and Sequential diagrams to illustrate the main purpose of the application.

1.3 Intended Users

The intended users of this document are software designers and developers. The software designer's benefit from this document as it assists them in keeping track architecture of the software as the application grows in the development phase. Moreover, software developer's can use this document to obtain a high level understanding of how the entities and features are interconnected within the architecture.

2 Software Architecture Overview

2.1 System Architecture Context

In order to understand the system architecture context, a very high-level diagram is shown in Figure 1. The users interact with our environment using different browsers. The user information for our application are taken from the Facebook public access APIs, this sets the scope of the user permissions, because users are allowed to interact and comment on other users activities only when they are friends. The users can login to the application from their browsers using this API and after login all the information related to the user regarding the app will be stored in a cloud infrastructure. For our front-end GUI we are using the Spring Framework (http://www.springsource.org/spring-framework).

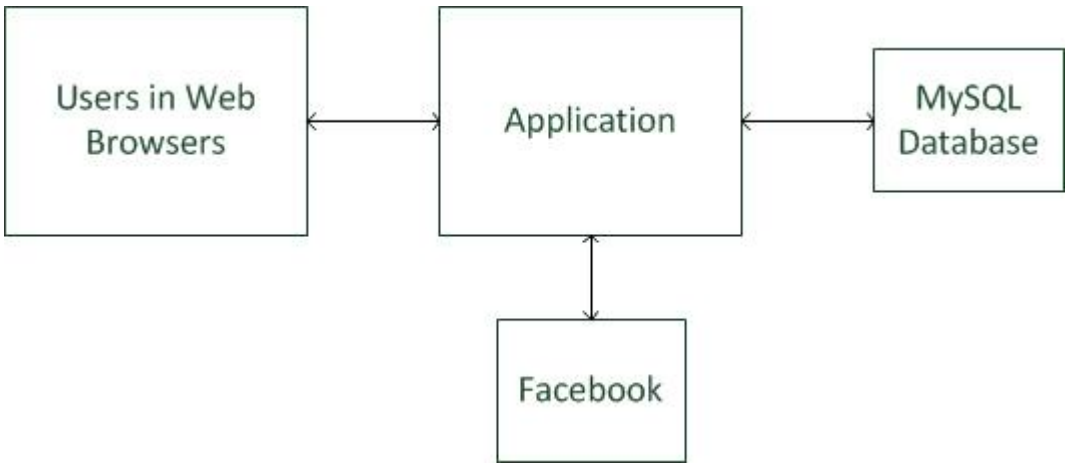

Figure 1: System Context Diagram

The cloud infrastructure will contain all the back-end servlets and JSP pages, which will serve to provide dynamic web content to the users. The package and class diagram for the application server is demonstrated later in this document.

The application will depend on a dynamic MySQL database to store user profile data and specific information related to the application. The database will also store an extensive library of calorie information for food items, which will be used to track the calorie consumption of users. This data is essential for other features. The database is normalized and handles concurrency to allow multiple users to interact in the application seamlessly. For hosting files the Glassfish Server is used.

A detailed representation of the physical entities involved in the system architecture is viewed in a deployment diagram in section 8 of the document.

2.2 Software Architecture Representation

This section illustrates the software architecture decomposition in terms of a package diagram.

2.2.1 High Level Decomposition

The package diagram shown in Figure 2 illustrates the high level decomposition of the software architecture followed by a description of each package and the connectors associated with those packages.

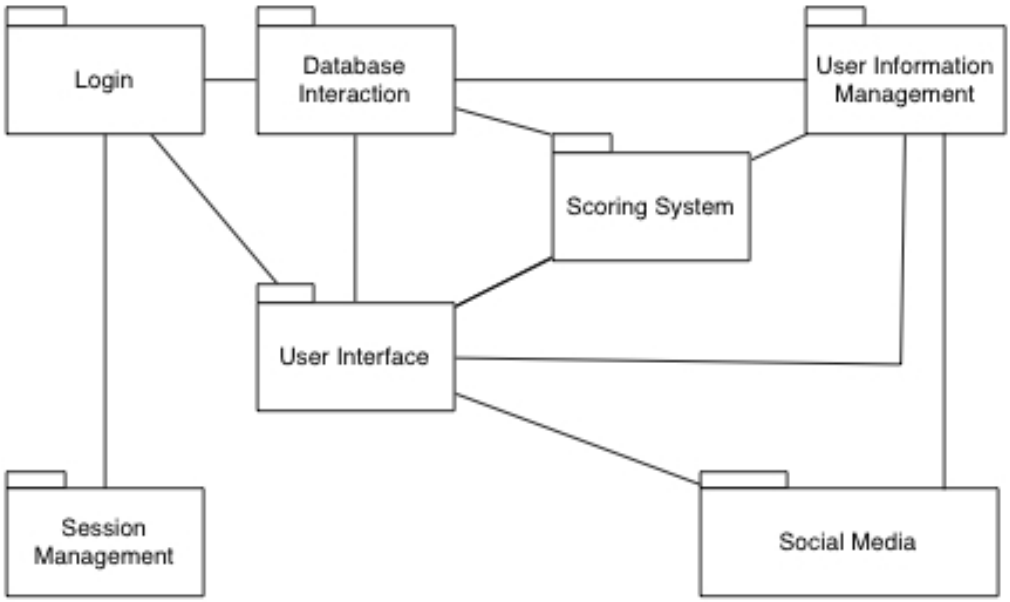

Figure 2: Package diagram for Healthy Eating Application

Session Management Package: a collection of classes associated with session management. The connector associated with the Session Management Package is with the Login package which stores the session activity of the user after the user logs in.

Login Package: a collection of classes related to gaining access to features available only to registered users. The Login package is connected is connected to the User Interface package by passing in the user information needed to update the interface. It is also connected to the Database package because of authentication process.

User Interface Package: a collection of classes that enable visual interaction with the user. It is connected to the Scoring System, Social Media and User Info Management packages to view information regarding those packages.

Social Media Management Package: a collection of classes related to the interaction of the application with Facebook. This package is connected to the User Info Management package because it is needed to make Posts of Facebook.

User Information Management Package: a collection of classes that handle user's personal account data such as their food entries, status and friend list.

Database Interaction: this package is related to database interaction operations. The Healthy Eating Application uses a considerable amount of different types of queries to the database thus a separate package for this activity is warranted.

Scoring System: a collection of classes associated with evaluating user food logs and assigning appropriate scores. This package also contains operations linked to determining if the user needs prompts for healthier eating choices.

2.2.2 Second Level of Decomposition

A second level of decomposition, identifying the main components within package, can be derived from Figure 1. This detailed decomposition highlights the major components present within the package. Note that components are units of computations and may be comprised of more than one class (marked as <<main component>> in diagrams). Other noteworthy packages are also shown.

The User Interface package contains the User Profile component; this is shown in Figure 3.

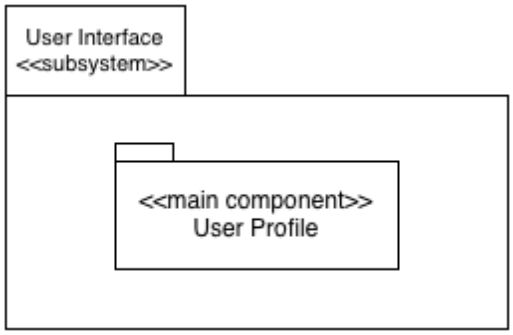

Figure 3: Decomposition of Data package

User Profile component: in accordance with the repository style there is one central data structure that represents the current state; for the Healthy Eating Application this is the user profile page. The user's profile will contain a list of all the user's food log entries for the current month (earlier entries can be seen by selecting a specific month and year). This component also has the option to display reports with graphical representation of user's food related data for archiving. Furthermore, the user's current score, and food optimization suggestions, friend's feedback for logged entries and social media elements are to be displayed and accessed from the profile. Since the user profile page and its feature are all based on the database relations, the data structure used to store the user profile in our case is the data base, as the content for each of the features are dynamically generated from tuples stored in the relational database. There are several independent components that operate on this central data structure.

The Database Interaction package contains two essential sub packages; these are shown in Figure 4.

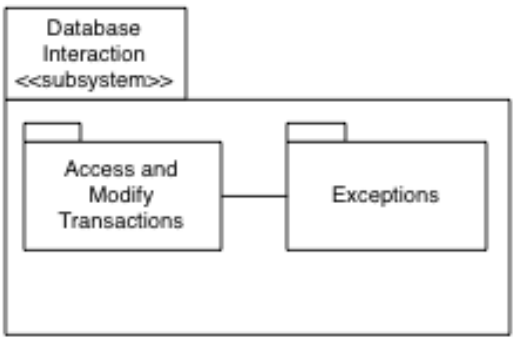

Figure 4: Decomposition of Database Interaction subsystem

Access and Modify Transactions: this sub-package is used by other main components in order to access and modify data in the database. It was not classified as a main component as it is better suited as a utility component.

Exceptions: this sub-package is to handle any erroneous modifications or accesses that the user might attempt. This does not provide any core functionality and thus was not classified as a main component.

The User Information Management package contains the several important components; these are shown in Figure 5.

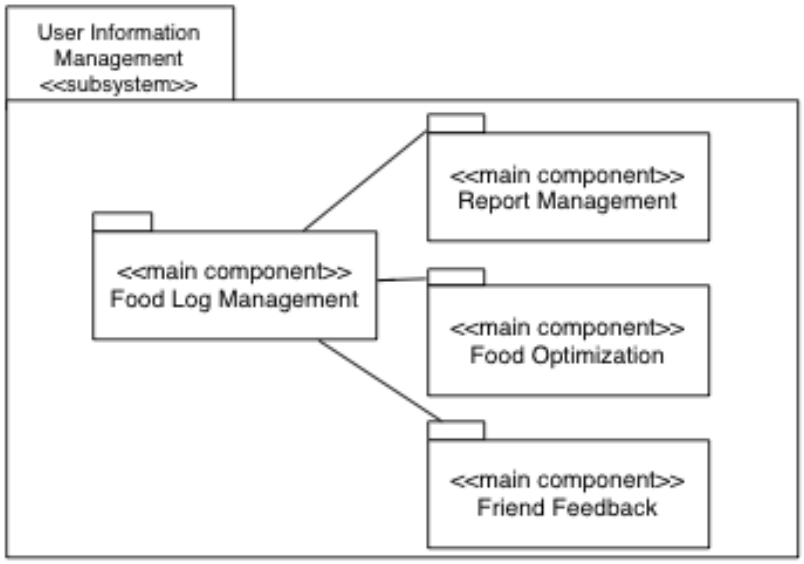

Figure 5: Decomposition of the User Information Management subsystem

Food Log Management component: the Food Log component process all food entry related data. It displays the first line of each entry on the user profile, with the option to view the entire entry if required. It is important to note that the user profile does not perform any action's of its own, it simply acts as a central body of information.

Report Management component: this component is responsible for processing all the calorie information in the entries present in the profile for the current month. The data is converted into different graphs that offer the user insight into their eating habits and patterns.

Food Optimization component: this component analyzes the user's food entries and statistics and suggests better food alternatives, if necessary.

Friend Feedback component: this component handles aspects related to feedback on the user's food log entries. The classes within this component ensure that only friends can comment, prevent any abusive and negative comments and display the comments properly.

The Scoring System package contains the Scoring System Management component; this is shown in Figure 6.

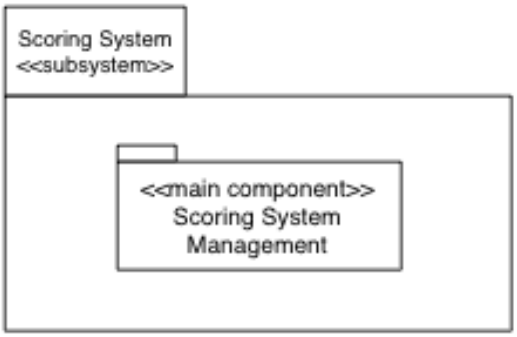

Figure 6: Decomposition of the Scoring System subsystem

Scoring System Management component: this component keeps track of the user's score, which is determined mainly by their food log entries.

The Session Management package contains the Cookies and Related Security Management sub-package; this is shown in Figure 7.

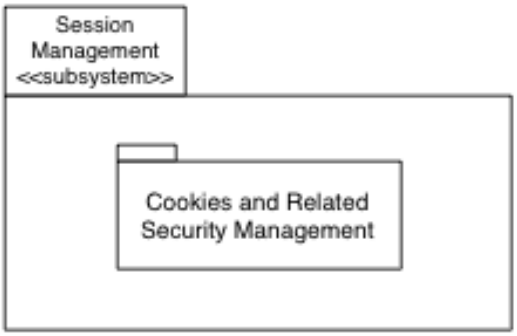

Figure 7: Decomposition of the Session Management subsystem

Cookies and Related Security Management: this package handles all cookie related operations such as saving certain session state information. It also contains any cookie confidentiality related code to preserve the cookie integrity.

The Login package contains the Login Management component; this is shown in Figure 8.

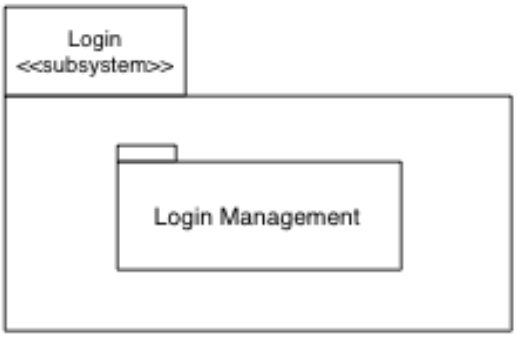

Figure 8: Decomposition of the Login subsystem

Login Management: This sub-package involves operation related to user login and providing critical features to improve the login security.

The Social Media package contains the Facebook Interaction Management component; this is shown in Figure 9.

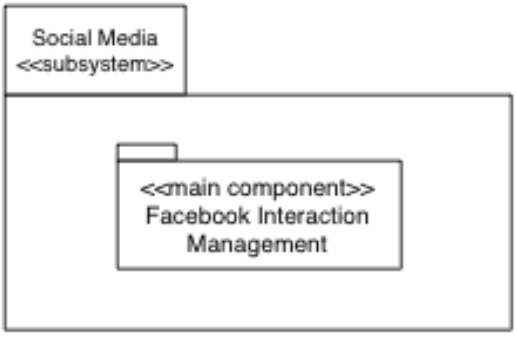

Figure 9: Decomposition of the Social Media subsystem

The Social Media Management component: this component is responsible for taking information reachable via the profile and displaying it on a relevant Facebook page as well as allowing the user to import certain application related data from Facebook.

3 Architectural Goals and Constraints

Different styles may seem more applicable depending on if one views the application solely from a software developer's perspective, user perspective or hardware implementation perspective. In general, the Healthy Eating Application is a three-layer software architecture since it consists of a user interface, middleware and a database system. However, it is important to note that in sections 3a and 3b, the style chosen was based on how the data would be presented to the user.

3.1 Architectural Rational and Justification of Style Chosen

The architectural style used is the repository style. Figure 10 illustrates how some of the major components and connectors present fit in to the architectural style chosen.

Figure 10: Components and connectors diagram for Healthy Eating Application

Justification of Style Selection

The repository architectural style is a justified selection for the Healthy Eating Application. This is because the user's profile is the central data structure. This is the case seeing as almost all major actions and activities affecting the application are displayed and accessed from there. What's more, any updates or modifications are displayed in some form on the profile. Additionally, the application lent itself to having a collection of independent components that operate on the central data structure. This was so, since most computations that occur in the application result in some form of change or update to the user's profile.

Architectural Rationale for Components and Connectors

It is important to mention the architectural rational for creating the different components. Each component encapsulates processing and data associated with a particular functionality within the application. For instance, the Food Log component encapsulates data related to a log entry and any processing associated with it. The Report component encapsulates data associated with generated reports and the accompanying computations. The same can be said for the Scoring System, Social Media, Food Optimization and Friend Feedback components. The rationale behind choosing user profile as the central data structure was that it houses snippets of all types of information provided by the application. Thus any changes made to certain type of data are reflected in some form in the user's profile.

Seeing as there is a central data structure in the repository style architecture, all the connectors are methods that link the content of the component with the user profile mainly via database queries.

3.2 Analysis of Architectural Elements Affecting Quality Aspects

Since there is a central data structure in the application, this gives the opportunity to have centralized management. In consequence, one can easily perform backups of essential data and focus mainly on the security of one central component. All of the aforementioned points increase the ease of maintaining the software.

In addition, the repository architecture is very efficient for sharing large amount of data. This is because all components have access to a common known data structure, which represents the central body of information.

However, a downside to the chosen architecture is that data evolution (defined in this report as integration of new type of information) might prove to be expensive. This is because the profile has limited space to show different types of data and displays. Therefore, the amount and type of data stored in the central data structure should be decided in the beginning, modifying this substantially in later stages could prove to be difficult.

4 Use Case View

The relevant use cases for the application are shown in Figure 11 and 12. The use cases have been divided according to if the user is logged in or not. Each figure is followed by the corresponding use case descriptions in tabular form.

Login: Use Case Description

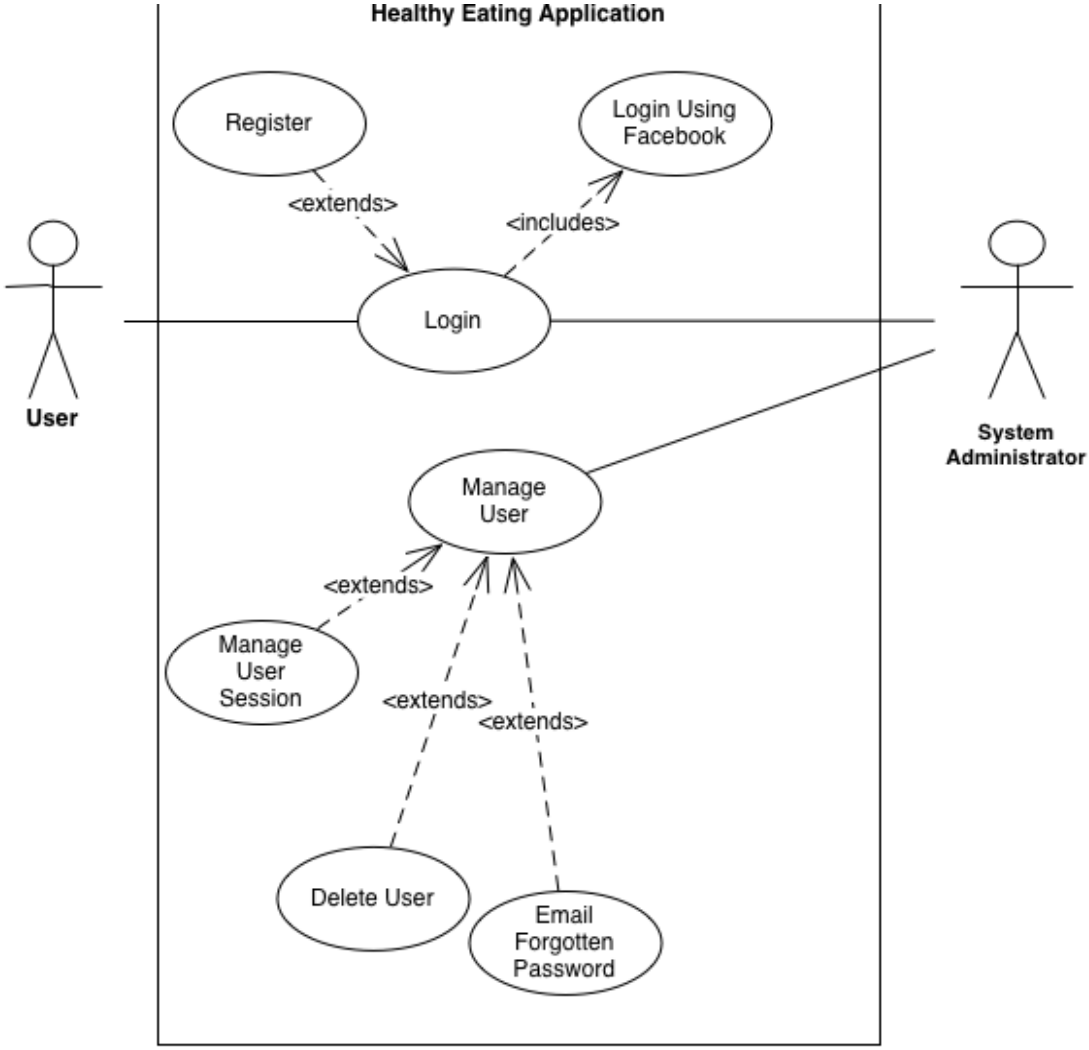

Figure 11: Use Cases concerned with logging into the application

Login	
Participating Actor	User
Entry Condition	User has correct credentials
Exit Condition	User is authenticated via database match
Flow of Events	User types ID/password and clicks on Login button. Credentials are passed to database to find a match. If hit, user exits this use case and is redirected to main page
Exceptional Cases	User's ID does not exist User's password is incorrect

Logged In: Use Case Description

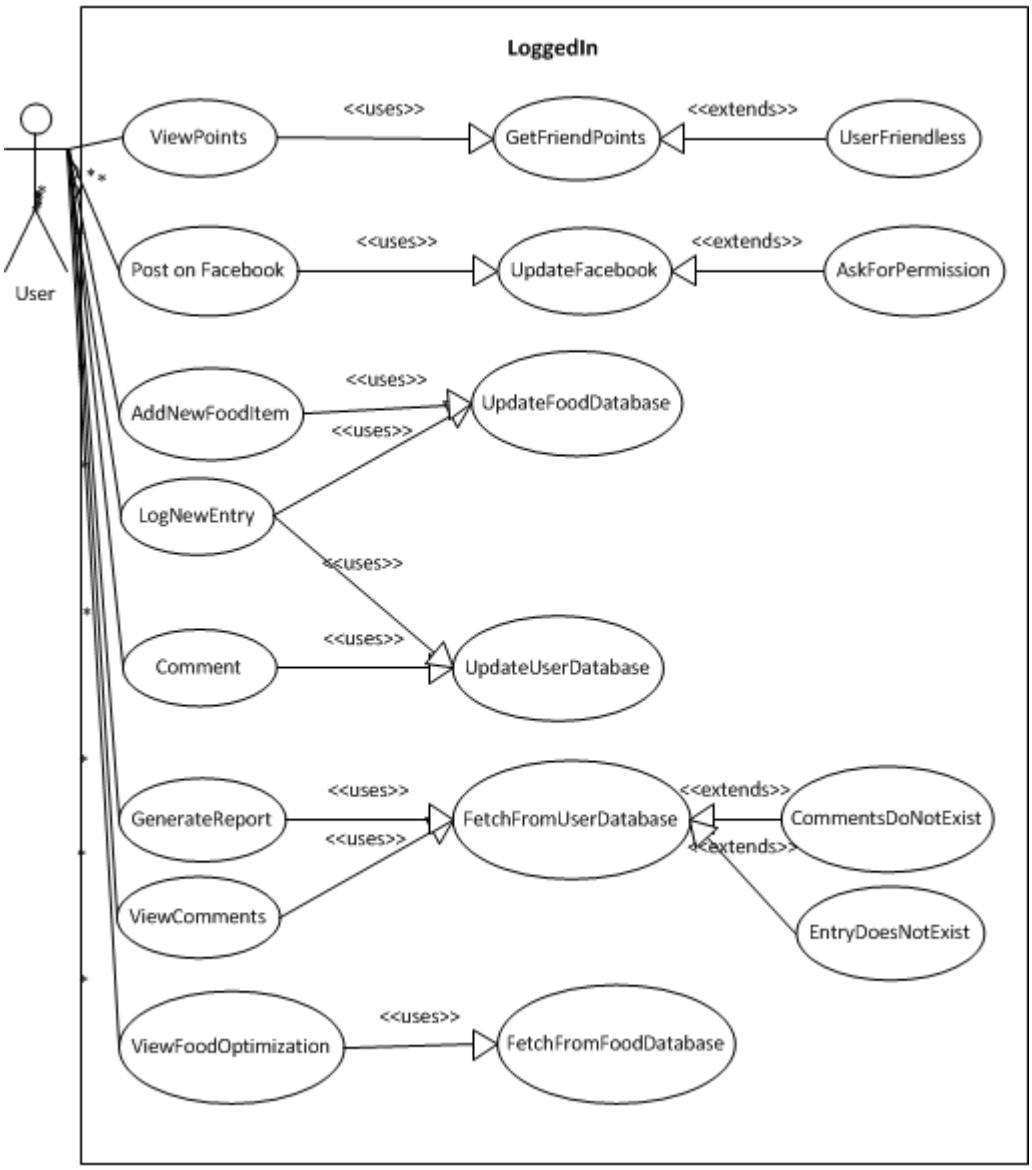

Figure 12: Use Cases concerned with when user is logged in to the application

View Points

Participating Actor	User
Entry Condition	User was authenticated via Login
Exit Condition	Points information is displayed
Flow of Events	User clicks on ViewPointBtn Database is queried with User's information Points information returned is displayed
Exceptional Cases	User does not have any past scores / friends

Post On Facebook

Participating Actor	User
Entry Condition	User was authenticated via Login
Exit Condition	Facebook status is updated
Flow of Events	User clicks on UpdateFBStatusBtn Database is queried with User's FB Credentials Facebook status is updated (changes coming to content of the update)
Exceptional Cases	User lacks content (to come later)

Add New Food Item

Participating Actor	User
Entry Condition	User was authenticated via Login
Exit Condition	New food item is added to the database
Flow of Events	User clicks on "AddNewFoodItem" button User fills out necessary fields(name, category, calories, etc) and clicks OK New entry is added to database accordingly
Exceptional Cases	Food item duplicate

Log New Entry

Participating Actor	User
Entry Condition	User was authenticated via Login
Exit Condition	New entry logged in UserDatabase
Flow of Events	User selects an item from the dropdown of food items from database User clicks Log New Entry button New entry is added to user database accordingly
Exceptional Cases	Intended food item does not exist in the database and hence does not show up in the dropdown

View Food Optimization

Participating Actor	User
Entry Condition	User was authenticated via Login
Exit Condition	Recommended food item is displayed
Flow of Events	User clicks on "Recommend food item" button User selects a category of food from dropdown Database is queried for most "healthy" choice for that food category Queried information is relayed back to the webpage and displayed
Exceptional Cases	User is too fat for any recommendations

Comment

Participating Actor	User
Entry Condition	User navigates to a friend's page
Exit Condition	New comment added
Flow of Events	User clicks on a friend's name from the list of friends User comments on an entry New entry is added to user database
Exceptional Cases	User does not have any friends

Generate Report

Participating Actor	User
Entry Condition	User was authenticated via Login
Exit Condition	Report is generated
Flow of Events	User specifies period and clicks on GenerateReport button Database is queried for points information for the period specified Points information is displayed via table or chart
Exceptional Cases	No data exists

View Comments

Participating Actor	User
Entry Condition	User was authenticated via Login
Exit Condition	Comments are displayed
Flow of Events	User clicks on any past entry from front page Database is queried for comments associated with that entry Front page repainted with the comment information underneath the entry
Exceptional Cases	No comments exist for the entry

5 Behavioural View

The behaviour of the system is illustrated using the sequence diagrams for some of the main use cases of the application. From among the use cases, the following are chosen for the behavioral view: *Log New Entry, Post on Facebook, View Points.* The *Log New Entry* use case illustrates how the activity sequence would typically look like for any new entry into the system. The *Post on Facebook* use case shows what the activity sequence would like for interacting with the Facebook API. The *View Points* use case illustrates how the activity sequence would look like for interaction between user profiles.

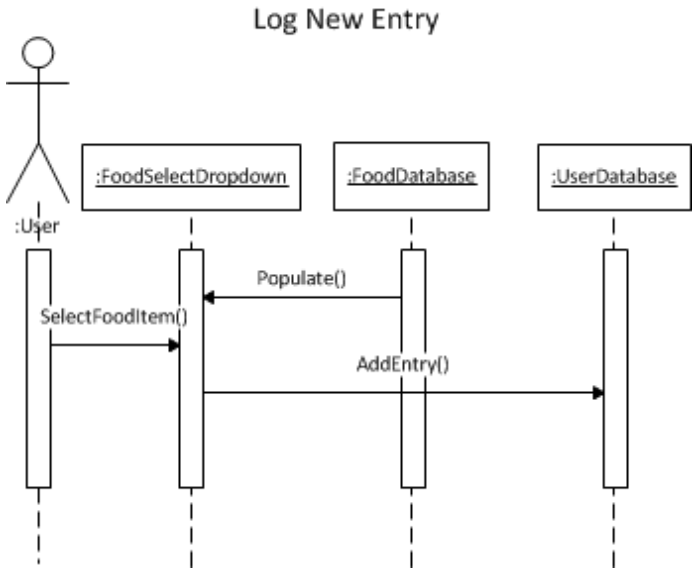

Figure 13: Sequential Diagram for Log New Entry Use Case

This diagram shown in Figure 13 illustrates how procedure calls are fired between objects when the user logs a new entry about his meal. From the front page, the user will interact with a UI feature, possibly a dropdown menu that is populated from the FoodDatabase object to select the correct food that was consumed. After selecting the food item, the user will trigger another UI feature that will fire the "SelectFoodItem" function, from which the Dropdown object will call "AddEntry" method with the parameter. The parameter will be the food item selected by the user. This will update the UserDatabase object accordingly.

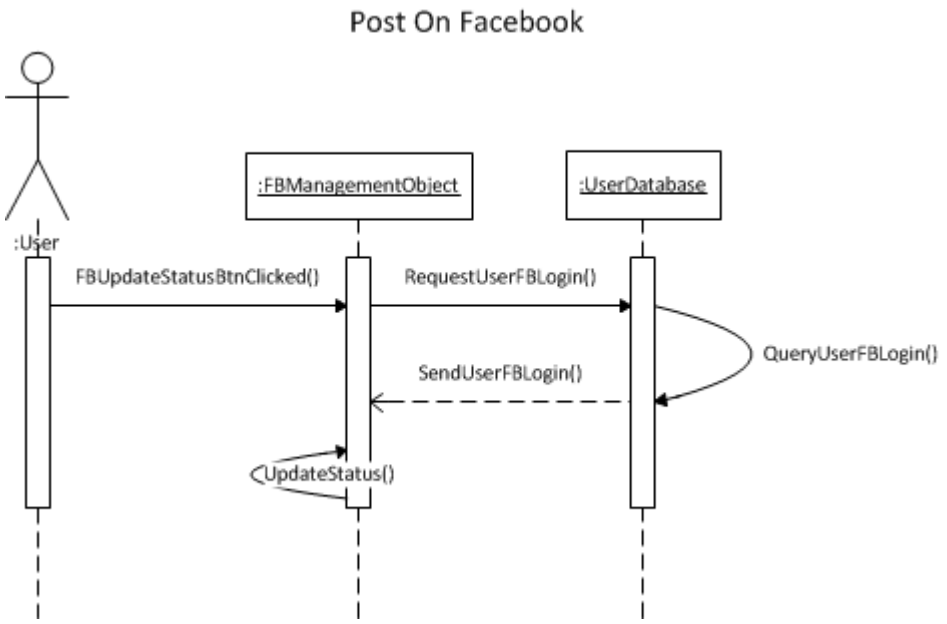

Figure 14: Sequence diagram for use case: Post on Facebook

The diagram shown in Figure 14 illustrates PostOnFacebook sequence of our application. The user will first click the button named "FBUpdateStatusBtn" which calls the FBManagement object. This object will then request user's FB Login information from the UserDatabase object. After the login information is received, the FBManagement object will call function "UpdateStatus" which will employ Facebook API to update user's status on Facebook.

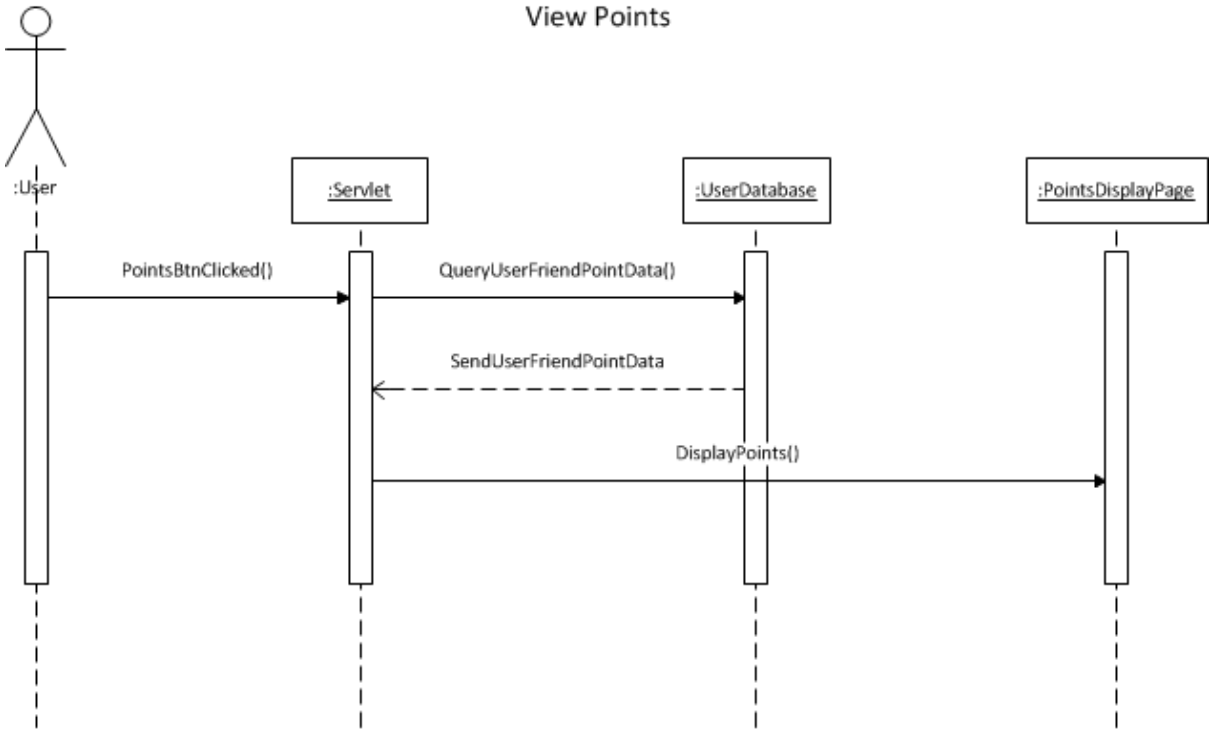

Figure 15: Sequence diagram for use case: View Points

This diagram shown in Figure 15 demonstrates the sequence for the user to view points that they scored. From the front page, user will click on a button named PointsBtn which will call the servlet object. Servlet will then query current user's point information from the UserDatabase object. After the servlet receives this information, it will redirect the user to "PointsDisplayPage," calling the function "DisplayPoints" which will paint a chart or a graph with the points information received from the database.

6 Viewpoint #3

There are a couple of concerns that are crucial to our architectural design which deal with run-time performance and concurrency and can be modeled as viewpoints. The first concern deals with the issue of many users attempting to log in to the application at the same time. This concern deals with concurrency, as several computations are executing simultaneously and potentially interacting with each other. To deal with this issue, the database implementation will use the transaction approach to solve concurrency issues. This means that at the start of a transaction the database will place a lock so that no other transaction can interrupt the process till the first transaction is completed. The database layer is interacted with using the Java servlets, which in turn interact with the presentation layer or the front-end. In addition, the Java threading support will be employed to further ensure that there is no concurrency issue with our software. Therefore, it allows a great way to deal with concurrency issues.

The other concern that had to be dealt with was the numerous potential remote calls that had to me made to servers to ensure the correct application of the software. This can also be modeled as a viewpoint by ensuring that only one back-end platform is used so all the calls are made to one server. This will be the main Java server that is being used for the development of the entire program. This concern deals with run-time performance issues as the remote calls could really slow down the program. This is essential to the design of our program, as we want to maximize the efficiency and performance.

7 Deployment View

The deployment diagram shown in Figure 16 describes how the application will be deployed across the hardware components.

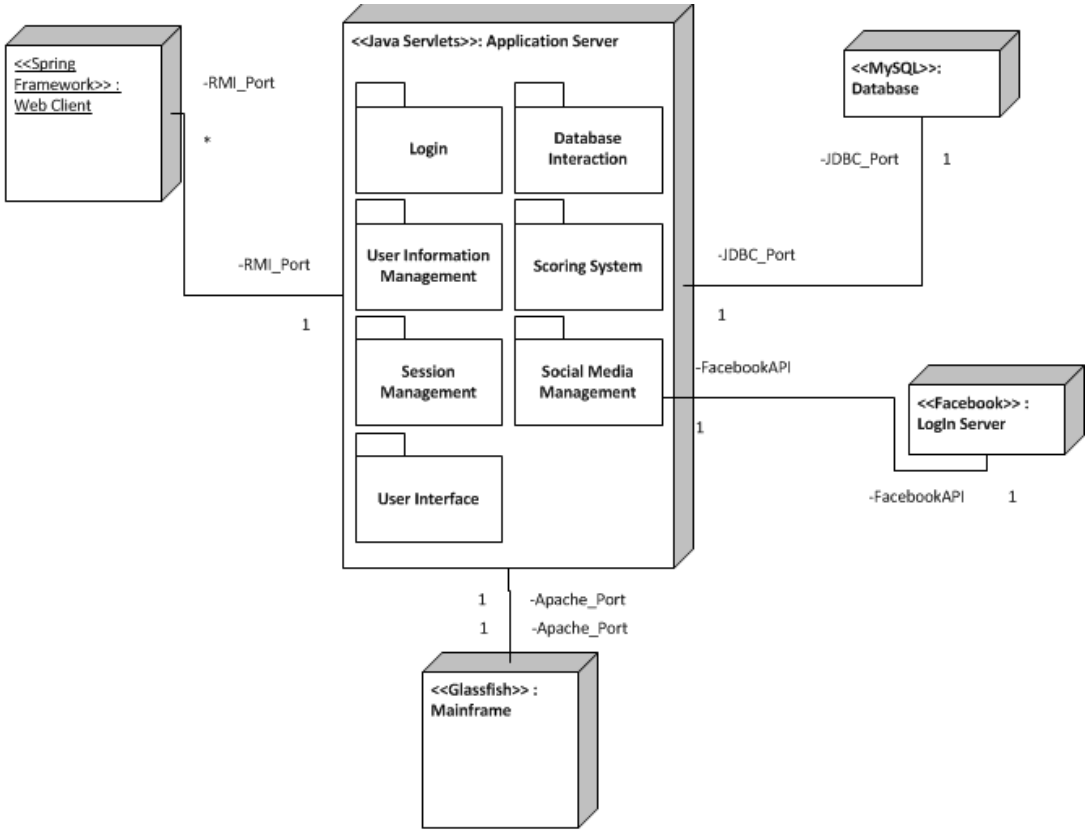

Figure 16: Application Deployment Diagram