Senior Design Project

S.U.N (Social University Network)

High Level Design Report

Group Members

Ahmet Ertan
Basri Karademir
Damla Özge Özgen
Dilan Yatan
Yeliz Kurt

Supervisor

Can Alkan

Jury Members

Hakan Ferhatosmanoğlu
Çiğdem Gündüz Demir

Innovation Expert

Mehmet Çakır

Table of Contents

1. Introduction
    1.1 Purpose of the system
    1.2 Design goals
    1.3 Definitions, acronyms, and abbreviations
    1.4 Overview
2. Proposed software architecture
    2.1 Overview
    2.2 Subsystem decomposition
        2.2.1 Presentation Tier
        2.2.2 Server Tier
        2.2.3 Logic Tier
        2.2.4 Data Tier
    2.3 Hardware/software mapping
    2.4 Persistent data management
    2.5 Access control and security
    2.6 Global software control
    2.7 Boundary conditions
        2.7.1 Initialization of Server
        2.7.2 Shutdown of Server
        2.7.3 Error on Server
        2.7.4 Initialization of Client
        2.7.5 Shutdown of Client
        2.7.6 Error on Client
3. Subsystem services
        3.1 Presentation Tier
        3.2 Server Tier
        3.3 Logic Tier

3.4 Data Tier

4. References

Table of Figures

Figure 1 - High Level System Architecture
Figure 2 - Deployment Diagram

1. Introduction

1.1 Purpose of the system

S.U.N app, aims to establish users to communicate each other during studying sessions or social activities in Universities. Rush of the school and concerning during the exams leads student to communicate lesser, and makes the situation harder for students because of the socializing problems. However, college environment is a better solution to socialize, but sometimes it is hard to encourage students to communicate people that they never met.

In order to consider these problems, S.U.N application aims to make people work together and participate events by meeting up by this application. Since, the technology helps us to reduce our concerns, this application will also encourage people to communicate when it is necessary and decorates the usage of the application with competitions and presents it as a way of having a joyful college life.

S.U.N app will be a system, which allows people to interact with each other during the studying sessions, or school events by challenging and beating the records. It aims to encourage people to socialize in spite of not being acquainted. Besides, people are able to see and write the comments of the lectures they have been registered. By attending events, meeting new people, working with them, especially creating own events leads users to earns some points for competition and challenging.

1.2 Design goals

Among our design goals there is ease of use, robustness, reliability and security.

Ease of Use: This is an app which should be designed with principles favoring ease of use. Since this is a user-driven app, it should be easy to use in order to maintain traffic through the app.

Robustness: This app should be robust enough to make a traffic with thousands of users. It shouldn’t crash if many people go online and use the app at the same time.

Reliability: S.U.N should be a reliable system in terms of exceptions, errors and crashes. Unexpected situations should be minimized.

Security: All the users for this system have private information such as planned activities, school information, passwords and mail addresses. So, this system should meet the security requirements in order to build trust with the users.

1.3 Definitions, acronyms, and abbreviations

MySQL: MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack. LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python. [1]

Java: It is a language that is developed by Sun Microsystems. Java is used in many computing platforms.[2]

NetBeans: NetBeans is an integrated development environment (IDE) for developing primarily with Java but also with other languages, in particular PHP, C/C++, and HTML5. It is also an application platform framework for Java desktop applications and others.[3]

Android: It is an operating system based on the linux kernel, and designed primarily for touch screen mobile devices such as smartphones and tablet computers. [4]

Four Tier Architecture: Three-tier architecture is a client–server architecture in which the user interface (presentation), functional progress logic ("business rules"),computer data storage and data access are developed and maintained as independent modules, most often on separate platforms. [5]

Android Studio: Android Studio is an integrated development environment (IDE) for the Android platform.[6]

ORM: It is a programming technique for converting data between incompatible type systems in object-oriented programming languages. This creates, in effect, a "virtual object database" that can be used from within the programming language.[7]

RDBMS: It is a database management system (DBMS) that is based on the relational model.[8]

REST: It is an abstraction of the architecture of the World Wide Web; more precisely, REST is an architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed hypermedia system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements.[9]

JSON: JavaScript Object Notation, is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is used primarily to transmit data between a server and web application, as an alternative to XML.[10]

1.4 Overview

While developing S.U.N app, we aim to implement it as a mobile application that provides a social communication platform. We aim to make the college life more fun for socialization of students who do not know each other before. The S.U.N app users can create events or activities or attend the events. The users allow to make comments for lectures or lecturers and share information about their common courses. Also, Facebook connection allows S.U.N app users to reach their friends and widen their number of friends to communicate much more enjoyable.

2. Proposed software architecture

2.1 Overview

There will be two main components of the S.U.N system: a backend component to answer the requests and a mobile application that will be a client to the backend. For this reason a client-server architecture is selected.

2.2 Subsystem decomposition

Aforementioned system components had three main responsibilities. There is a component that keeps the data for the system, a component that manages the application logic, a component, which listens to the requests coming from the mobile application, and finally a presentation component. For this reasons the Four-Tier architecture was a logical choice.

On Four-Tier architecture every layer acts a client to the layer below and as a server to the layer above. This approach is very suitable for S.U.N since the presentation layer is a remote layer and cannot accept requests from the logic layer. Moreover, Four-Tier architecture also reduces the coupling between the components, which is also an important for the system.

Backend of the system will contain the data and the logic layers of the system while the mobile application is only responsible for the presentation. However, Android system architecture encourages Model-View-Controller architectural pattern. Hence, the presentation layer of the S.U.N is also divided into three components.

Model part of the mobile application is responsible for fetching the data from the server and sending the updates on the data. Moreover, this part also contains the visible representation of the data classes. Since the server will only send the relevant information to the mobile application, data representation on the mobile application will be similar to the actual data on the server but with some missing or additional information.

View part of the application will be implemented using the Interface Builder tool of Android Developer Toolkit. IB is a WYSIWYG interface editor, which separates view and its controller completely.

Finally there is the controller part of the mobile application. Controller will handle the user's action on the views and manage mobile application's logic.

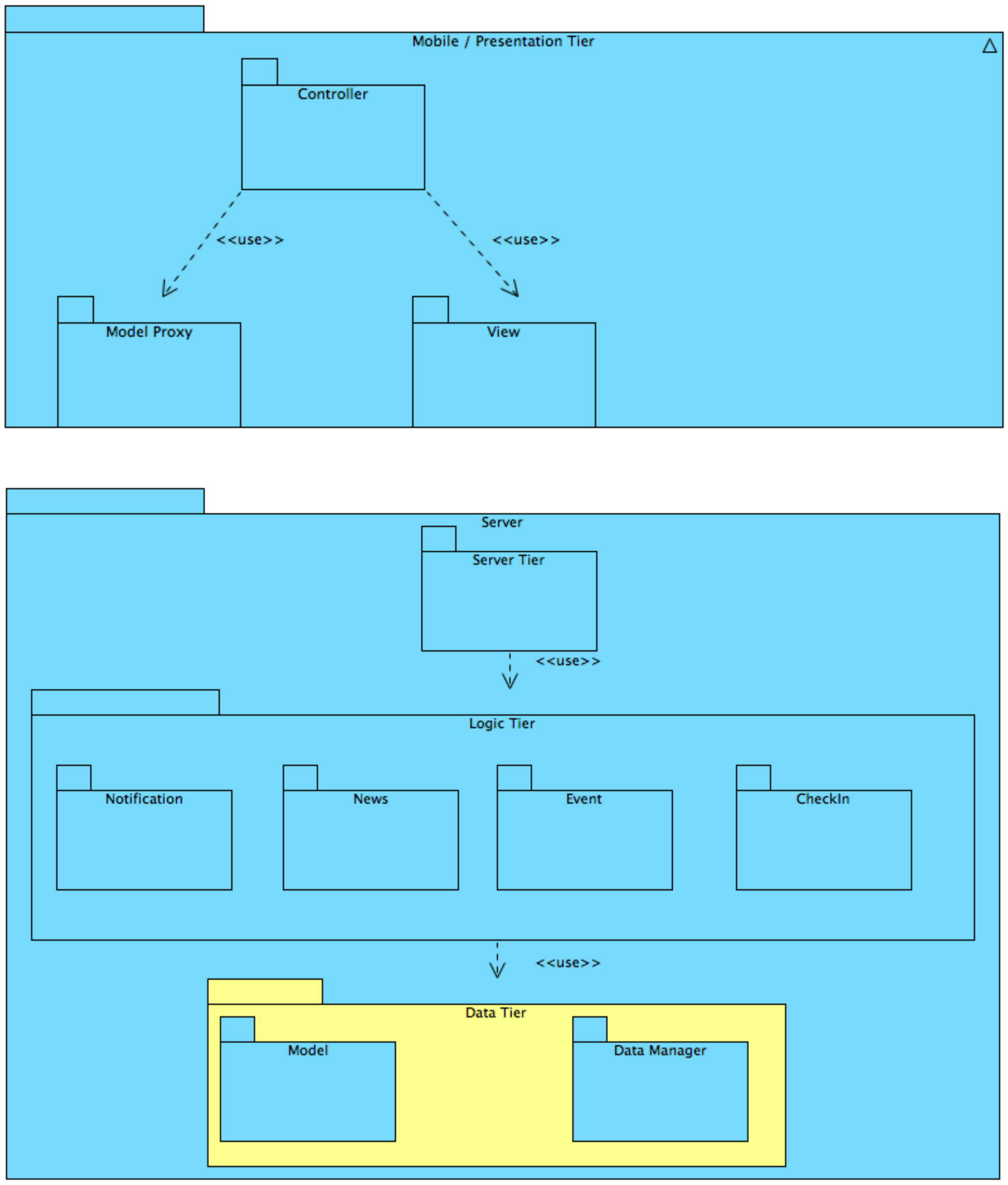

Figure 1 - High Level System Architecture

2.2.1 Presentation Tier

Presentation tier is the client of the SUN system, which is the Android application. It has three main components described below.

2.2.1.1 Model Proxy

Responsibility of this subsystem is to be the representations of the model objects of the server component. This part only keeps data for the client.

2.2.1.2 View

This subsystem manages the GUI part of the mobile application.

2.2.1.3 Controller

Controller subsystem manages the mobile application; it handles the data communication between client and server and handles user events.

2.2.2 Server Tier

This part is responsible for listening for the client requests and delegating the requests to the layer below.

2.2.3 Logic Tier

Logic tier manages all application logic; it checks permissions of the users, creates necessary objects and monitors the relationships between the system objects.

2.2.3.1 Notification

This subsystem decides when a user should be notified and sends necessary notifications to the mobile devices of the users.

2.2.3.2 News

News subsystem is responsible for displaying the relevant news to a certain user.

2.2.3.3 Event

Event creation process takes place in this subsystem.

2.2.3.4 CheckIn

Decides whether or not a user can check-in to an event and number of points that will be awarded to the user.

2.2.4 Data Tier

Data tier manages all persistent data storage activities that is needed by the upper levels.

2.2.4.1 Model

Keeps the model objects, which keep the data of the system.

2.2.4.2 Data Manager

Data Manager provides an API for the upper levels, it creates and handles required objects for the ORM tool.

2.3 Hardware/software mapping

Three main parts of the system will be deployed to three hardware components. There will be indefinite number of Android devices that run the mobile application. An application server will be deployed on a Linux machine, which has at least 1gigabayts of memory and 20gigabayts of flash storage. Finally an RDBMS server will run on a dedicated machine, which also has the same specifications with application server.

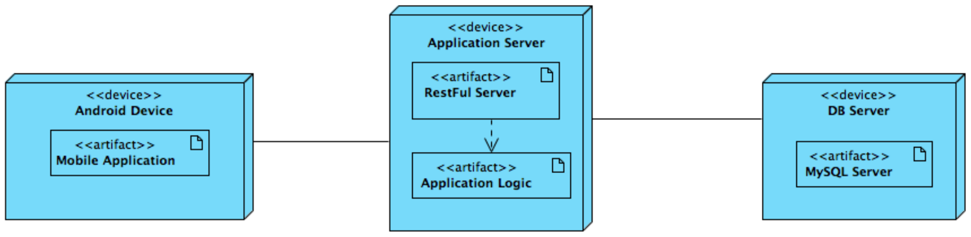

Figure 2 - Deployment Diagram

2.4 Persistent data management

All persistent data will be stored in a relational database using an ORM solution. Due to the out of the box performance, scalability and reliability benefits Hibernate was chosen as the ORM manager.

After an early analysis, it was seen that MySQL satisfies the project requirements and can be considered as the primary choice. However, Hibernate supports the majority of the modern RDBMSs and migrating to another one without losing any data is trivial.

2.5 Access control and security

Every S.U.N user has a username and password pair for identification. When the user wants to use the mobile application, the correct username and password pair must be provided. A token-based authentication system will be utilized for authentication and authorization purposes.

As the first step of the authentication, API client of the mobile application will send the credentials of the user to the login end point of the API. Login system will perform the checks and generate a token for the user if the entered information is correct. Any further requests to the API will contain the authentication token in the HTTP header. Backend of the application will match the authentication token and respond the request.

While responding the request, the backend will honor the authorization information of the data. Private information about users or groups will not be included in the responses given to unauthorized users.

There will be benefits of using a token-based authentication over Basic HTTP Authentication or a similar strategy that sends username and password with each request. These benefits are, shrinking the window of opportunity of capturing the password, eliminating the need of saving the credentials of the user to the mobile device, easy revocation of authentication tokens.

In order to implement an auto login feature on the mobile application, authentication information must be kept on the mobile device. In case of the username and password based system, the authentication information will contain the personal password of the users and using a token-based system will prevent this.

Furthermore, if an attacker somehow captures the authentication token, generating a new token will revoke the captured one and the captured token will be useless.

The passwords of the users will be kept on the database by salting and hashing. Salts of the users will be unique and be generated using a secure random number generator. Then salt will be prepended to the user’s password and hashed with SHA256 hashing algorithm.

Finally, all data communication between mobile application and the backend will use HTTPS protocol for safety reasons.

2.6 Global software control

Server side of the application waits for the incoming connections on the main thread. When a client arrives, server creates a new thread and starts serving the client on new thread so that it can continue serving new clients. After the creation, new thread starts listening for the requests.

The control flow on the mobile application is mainly event driven. There are two kinds of events that can occur on the application; user generated events and notification event.

Users of the application generate events when they interact with the application, click on the application or system buttons, etc. Android architecture allows developers to register listener for these events, then controller subsystem of the application handles the operation.

Notification events arrive at the mobile application via GCM, and Google manages the delivery of these events. After the event arrives at the mobile device, SUN app will perform necessary steps to notify the user.

Any data communication between client and the server will happen in a background thread in order to provide a better user experience.

2.7 Boundary conditions

2.7.1 Initialization of Server

In order to initialize, server components must connect to the database server. Then the component, which is responsible for handling client requests, must start listening to 443th port for HTTPS connections.

Server components do not keep any client state information in volatile storage. Thus there is no need to fetch data from the database or any other preparations.

2.7.2 Shutdown of Server

Only necessary action while shutting down the server is to close the open network connections with the clients.

2.7.3 Error on Server

Any exception that thrown on the server must be caught and the server must respond with meaningful error codes instead of *500 – Internal Server Error.*

2.7.4 Initialization of Client

Client needs an active internet connection so as to start working. If the internet connection is available, ApiClient component of the application initializes. It prepares the connection between client and the server.

If the mobile application has a saved token for the user, then it fetches the personal information using this authentication token. This also validates the token.

2.7.5 Shutdown of Client

The user of the mobile application may choose to log out by pressing the logout button on the settings. In that condition, the application deletes the saved authentication token from the mobile phone’s persistent data storage.

The user may also want to shutdown the application. This action cancels all active requests and the application sends any unsynchronized data to the server in background.

2.7.6 Error on Client

Any thrown exception should be caught and a descriptive error message should be displayed to the user. If the errors are recoverable automatically, then the client should try to recover from these errors without disturbing the user.

3. Subsystem services

3.1 Presentation Tier

Presentation tier is the client of the SUN system, which is the Android application. It has three main components described below.

3.1.1 Model Proxy

This subsystem keeps the model objects of the system. These classes are proxies for the model objects of the backend. Even though the model objects are consistent with their referring objects, they have only the necessary information for operations, since the mobile application does not have to know all information to work.

3.1.2 View

As mentioned above, the view subsystem of the client side will be implemented using interface builder of the Android Toolkit.

3.1.3 Controller

Controller part of the presentation tier manages the logic of the mobile application. This includes navigation between pages, operations and communication between the mobile application and the server.

APIClient is also a part of this subsystem. It prepares and sends requests to the server using Rest principle. Data passes between client and server using JSON serialization.

3.2 Server Tier

This part is responsible for listening for the client requests and delegating the requests to the layer below. This tier provides the upper layer, client, with a RestFul Api. Play! Framework will be used to implement the server tier. This tier delegates the requests to the layer below and returns the results to clients. Necessary URL routes for each resource will be defined using the Play! Framework’s router.

3.3 Logic Tier

This tier manages the all logic behind the application. It checks authorizations, states of the objects and communication between them. There is a Façade class for this tier, which delegates the requests coming from the server tier to the sub-subsystems. There are four subsystems of this component. These are *notification, news, event* and *check-in* subsystems. They are highly cohesive packages that perform only their respective jobs without a coupling between them.

3.4 Data Tier

3.4.1 Model

This tier keeps the model objects. These objects also implement the requirements of the ORM manager. Hibernate requires model objects to describe their fields, table names and any constraints.

3.4.2 Data Manager

Data Manager handles the communication between the system and the RDBMS. It provides a simple interface to the upper level, which is described by an abstract class, DatabaseManager.

4. References

[1] PHP. Wikipedia. Nov 3, 2013

http://en.wikipedia.org/wiki/PHP

[2] Java (software platform). Wikipedia. Nov 3, 2013.

http://en.wikipedia.org/wiki/Java_(software_platform)

[3] NetBeans. Wikipedia. Dec 23, 2014

http://en.wikipedia.org/wiki/NetBeans

[4] Android. (n.d.). In Wikipeida. Retrieved Dec 25, 2013, from

http://en.wikipedia.org/wiki/Android_%28operating_system%29

[5] Multitier Architecture. Wikipedia. Dec 24, 2014

http://en.wikipedia.org/wiki/Multitier_architecture#Three-tier_architecture

[6]Android Studio. Wikipedia. Dec 24,2014

http://en.wikipedia.org/wiki/Android_Studio

[7] Object-Relational Mapping. Wikipedia.Dec 4, 2014

http://en.wikipedia.org/wiki/Object-relational_mapping

[8] Relational Databae Management System. Wikipedia. Dec 27,2014

http://en.wikipedia.org/wiki/Relational_database_management_system

[9] Representational State Transfer. Wikipedia

http://en.wikipedia.org/wiki/Representational_state_transfer

[10] JSON. Wikipedia. Dec 25, 2014

http://en.wikipedia.org/wiki/JSON