TED UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

CMPE 491 – Senior Project Report

High-Level Design Report

Name: Çağla KÖSE, Doğa TÜRKMEN, Mert ARCAN, Oytun Uras ŞAHİN

Supervisor: Venera ADANOVA

Jury Members: Aslı GENÇTAV, Elif KURTARAN ÖZBUDAK

Table of Contents

1. Introduction

Univhex aims to increase the amount of interaction between students on the same campus. Students will be able to register on this platform by verifying their university e-mail addresses. This way, we ensure that non-students will not be able to register to the app. The app structure will have a similar structure to other social media apps in today's world. It will have a profile page for every individual user where the information about them takes place. A home page where users shared content, such as posts, will flow. Additionally, it will have a Univhex page that has some of the most-liked posts will take a place. The purpose of this page is to increase the ratio of the post's attraction. To get into this page, posts should get some amount of "Hex's", which will be a button similar to comments and likes, but in a hexagon shape. After getting enough Hexes that post will automatically transfer to the most-liked page. On Univhex page, the posts keep their position there for at least a week. Sharing a post in Univhex will be able to be done anonymously or with the user's name. Furthermore, Univhex will comprise a rewarding system to keep users interacting with the app. This rewarding mechanism will be related to earned hexes with posts. Finally, we need to inform that, Univhex will be a social interaction platform and it won't contain any educational purposes.

1.1 Purpose of the System

The Univhex is a social media app for university students. The main purpose of our app is to make it easier for students to communicate with each other. With the help of our app, students will be able to share their thoughts about school and ask questions inside the app. Students will be able to do these anonymously or by their names. Univhex aims to enhance the relationship between students in the same university.

1.2 Design Goals

The main aim of Univhex is to level up the communication between students. We have chosen to incorporate the following characteristics into our software architecture to create a functioning and extendable system:

1.2.1 Extensibility and Modularity

We are planning to make our system to be highly modular. Any other modules will be incorporated without interfering with existing ones. The system will be in such a way that adding other modules will be simple, provided that the module is created.

1.2.2 Maintainability

This application will be designed to implement possible future upgrades easily. Methods in the code will include comments to describe what is the main purpose of that method to help us understand each other’s work in the future.

1.2.3 Reusability

Any module of the project can be easily separated and utilized in other applications because our system is extremely modular and build to function as a pipeline. Since our project is to make a social media app any module could be used in other social media projects as well.

1.2.4 Usability

The interfaces will be designed in a user-friendly way so users can access the app without needing any extra explanation while performing operations on the application.

1.2.5 Performance

Univhex will have a bit of a processing delay while uploading and updating the main page. We are aiming to keep the delay in the possible minimum range to not lose users by inactivity.

1.2.6 Availability

Unless there are unexpected system failures, the Univhex will be available and reachable 24 hours a day and 7 days a week.

1.2.7 Scalability

We are planning to add new universities to our system when we see a demand from the students. Rapid action is required to meet a rise in demand so that there is no drop in performance or system outages.

1.3 Definitions, Acronyms, and Abbreviations

Social Media: Websites and applications that enable users to create and share content or to participate in social networking.

Post: The user uploading their questions or thoughts to the system.

Like: The user interacting with the posts they agree with.

Comment: The user giving their opinions on.

Client: The application interacting with the user.

Server: The system communicating with each client and update their page.

1.4 Overview

Univhex is a social media application, that aims to increase the interaction between university students on the same campus, meaning that only the students at the same university will be able to reach each other's content. Univhex will be available for both android and IOS devices. The user interface structure of Univhex will be similar to today's social media platforms. It will have a login and register systems, where users can sign up or sign into their accounts. Besides, it will have a profile, home, search, and Hexed posts (Name is not specified yet) screen, where users will spend most of their time actively in the app. The profile page will include general information about the user such as name, surname, the field of major, profile photo, grade as year, and previous posts he/she shared. The home page will be the mainstream page where each user's post will be displayed. The search page will be the field where you can search other user’s profiles. Lastly, hexed posts (Name not specified yet) will display the posts which are liked by the community. But how will this work? The concept of Univhex will be about hexes. Users will be able to increase their hex stats by making interactions within Univhex. As a simple example, if I share a post and it gets the required amount, currently we are thinking that 6 is enough, of like’s it will be awarded with some amount of hexes, and users who pressed the like button for the post also will be awarded with some amount of hexes, which will be much lower than the post owner. In conclusion, Univhex is a social media app built for increasing the interaction between students, which will be a unique app for university students. Our team is developing the project with agile methodologies. So, we are brainstorming and adding new features and extracting the old ones continuously.

2. Proposed Software Architecture

2.1 Overview

This part will share more information about Univhex’s structure which were the methods and features mentioned in the Analysis report. In the upcoming subsections, the paper will discuss the subsystem influences, modules that are done or in progress, and the systematic resolutions that we made up to this point.

2.2 Subsystem Decomposition

The Univhex project’s architecture will be based kind of on a client-server model, but this project uses Firebase services as a server. That means the Univhex project does not have a traditional server because Firebase itself does not define it as a server. But on the other hand, this service does everything that a server does. The client subsystem will be a mobile application that allows the client to register and log in to the application, share posts on the home page, share posts anonymously, hex (our like system) posts, comment on posts, upload photos for their profile, and search other user’s profile. On the other hand, Firebase service, which is our server, copes with Univhex’s data processing while users share posts or give hexes to other posts. Also, it will store user data, like posts or likes, on the cloud as well as authentication information.

The client-side is separated into two main subsystems: the Presentation subsystem, and the controller subsystem. The presentation subsystem would handle the graphical user interface in which the user interacts. It is divided into different modules for viewing screens and handling requests that would be created by user interactions. On this subsystem, Univhex has all the view models such as LoginView, RegisterView, ProfileView, MainstreamView, UnivhexView, SearchView, and CreatePostView. Each of these views has a unique view for the user. For example, when a user wants to see his/her profile, ProfileView communicates with the controller which is the request manager and waits for the response. When the response comes, it will show the profile page.

The server side is divided into three parts. These parts are user manager, content manager, and request handling. The user manager handles login and registration for each user and has a list that contains every user's information. If there are any changes or modifications to a user, the user manager will control and authorize it. The content manager checks the posts that users share and stores the data of it. For example, when a user shares a post, the content manager stores this post as a user post with its content. Finally, request handling communicates between clients. For instance, if the user wants to share a post request handler verifies the user from Firebase and then sends it to the content manager.

2.3 Hardware/Software Mapping

Univhex's server-side modules and database management will be done in Firebase service and Firestore. This service communicates through Firebase Database API which is a service of Firebase. The client will create an HTTP request depending on its actions and Firebase returns a suitable response. The client can reach the application from every Android and IOS device. All the features will be worked on both kinds of devices (IOS and Android). To sum up, the client and the server (Firebase) will connect through HTTP request and Firebase Database API.

2.4 Persistent Data Management

Persistent data includes user credentials, posts, and profile information of the users. User credentials are hashed and stored in Firebase and Firestore. Additionally, credentials are also stored in the user's local device to authenticate the user directly while the app is launching. So, the user will not need to enter credential information to sign in. Posts and user information will be stored in Firestore. Objects will be stored in the Firestore in JSON format. However, images, such as profile pictures, of users can be stored in Firebase storage to lead to more of a cost-effective storage.

2.5 Access Control and Security

Users must create an account to use Univhex. To create an account, the user is required to select an email and password. Email is required to contain a '.edu' extension as proof that the user is currently a member of the university. The user account can share new posts, see posts shared by other students who are members of the same university, and profiles of those students. Passwords of users will be stored in Firestore and local devices after a hash operation. Firebase will never acquire the passwords as plaintext.

All the services used by Univhex are provided by Firebase. Our authentication system is Firebase Authentication which integrates tightly with other Firebase services and leverages industry standards like OAuth 2.0 and OpenID Connect, so it can be integrated easily with our custom backend.

.

2.6 Global Software Control

Univhex uses an event-driven software control. Whenever, the user creates a new post, interacts with another user's post, or changes their profile information, the system sends requests to firebase. The Firebase request handler will take this request and returns the processed result to the client. This control mechanism is also applied to other aspects such as authentication where the user signs in to send a new request to firebase which will return as an authentication token. After log in, the system will send a new request to Firestore to fetch the latest posts, again by a request.

2.7 Boundary Conditions

2.7.1 Initialization

To use the app, users need to access the Univhex client. The client is usable through tablets and smartphones that use either IOS or Android operating system. Firstly, the user needs to be authenticated by their university email to sign up and this process is handled by using the client. In order to sign in they need to enter their e-mail address and their password. The user is taken back to the login screen if the login credentials don't match what was anticipated and authentication fails. An internet connection is necessary at this point since the authentication happens on the Univhex server. When the user logs in to the Univhex, he/she is directed to the main page.

2.7.2 Termination

Except for times when the client is waiting the outcome of an operation, the user can log out of Univhex whenever they want.

2.7.3 Failure

Client-related data is attempted to be saved before termination in the event of a crash in the client. Because most of Univhex's functions must be used with internet connection, we expect most of the problems to be the result of a lack of an internet connection.

3. Subsystem Services

3.1 Client

3.1.1 Presentation

The presentation layer that contains UI views and their event firing mechanisms.

UIManager: A global manager to moderate other view classes.

LoginView: The view class for the login screen. Validates credential inputs before request creation. Uses RequestManager to make login authentication requests.

RegisterView: The view class for the register screen. Validates user credential inputs before creating a request. Uses RequestManager to make register requests.

ProfileView: The view class for user’s profile screen. Displays necessary user information on the page, including name, surname, university name, field, year of study, and previous posts posted by the user. Uses RequestManager to make profile edit requests.

MainStreamView: The view class that displays main-stream (home) screen where user can see posts posted by the same university members. Uses RequestManager to request main-stream page data.

UnivhexView: The view class that displays Univhex posts. Uses RequestManager to fetch Univhex posts data.

SearchView: The view class which displays a search-bar at top, and related users according to input of search-bar field. Uses RequestManager to request corresponding users data.

CreatePostView: The view class which displays the screen for user to create a post. Uses RequestManager to create an add new post request.

3.1.2 Controller

Controller is responsible for listening client events to establish any client-server communication, local storage, and request creation.

RequestManager: Creates all the requests depending on the events fired by presentation layer and establishes the communication with server. Awaits the server response and notify necessary views based on the response.

LocalStorageManager: Is responsible for the storage and management of certain types of data on the user's device, such as cached content or user preferences.

3.2 Server

3.2.1 Request Handler

Request Handler is a subsystem that is responsible for processing requests made by the presentation layer to the server-side. These requests could include requests for data, requests to perform an action, or requests for authentication.

3.2.2 User Manager

The server-side infrastructure would handle the creation and management of user accounts, as well as the process of verifying a user's identity when they log in.

3.2.3 Content Manager

Responsible for handling the storage and organization of content, as well as handle interactions between users.

4. Glossary

Agile – a project management approach based on delivering requirements iteratively and incrementally throughout the life cycle. Agile development – an umbrella term specifically for iterative software development methodologies. Popular methods include Scrum, Lean, DSDM and eXtreme Programming (XP).

Database – a collection of information related to a particular topic or purpose. There are two types of databases: Nonrelational and relational. Database management system – a program such as Access, that stores, retrieves, arranges, and formats information contained in a database.

Hashing –hashing in the data structure is a technique of mapping a large chunk of data into small tables using a hashing function.

Framework – a layered structure indicating what kind of programs can or should be built and how they would interrelate. Some computer system frameworks also include actual programs, specify programming interfaces, or offer programming tools for using the frameworks.

Portability – a software characteristic, portability is a form of reusability that helps to avoid "lock-in" to certain operating environments, e.g. cloud providers, operating systems or vendors.

5. References

- https://firebase.google.com/docs/auth
- https://www.simscale.com/docs/simwiki/numerics-background/what-are-boundary-conditions/
- https://stackoverflow.com/questions/33825967/the-subsystem-and-the-components-of-the-system-in-software-engineering