TED UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

CMPE 492 – Senior Project Report

Low Level Design Report

Project Title: Univhex

Group Members: Çağla KÖSE, Doğa TÜRKMEN, Mert ARCAN, Oytun Uras ŞAHİN

Project Advisor: Venera ADANOVA

Jury Members: Aslı GENÇTAV, Elif KURTARAN ÖZBUDAK

1.Introduction

Univhex points to extend the sum of interaction between understudies on the same campus. Understudies will be able to enroll on this stage by confirming their college email addresses. This way, we guarantee that non-students will not be able to enlist to the app. The app structure will have a comparable structure to other social media apps in today's world. It'll have a profile page for each person client where the data almost them takes put. A domestic page where clients shared content, such as posts, will stream. Also, it'll have a Univhex page that has a few of the most-liked posts will take a put. The reason of this page is to extend the proportion of the post's attraction. To urge into this page, posts ought to get a few sum of “Hex's”, which can be a button comparative to comments and likes, but in a hexagon shape. After getting sufficient Hexes that post will naturally exchange to the most-liked page. On Univhex page, the posts keep their position there for at slightest a week. Sharing a post in Univhex will be able to be done namelessly or with the user's title. Besides, Univhex will include a fulfilling framework to keep clients interacting with the app. This fulfilling component will be related to earned hexes with posts. At long last, we got to advise that, Univhex will be a social interaction stage and it won't contain any instructive purposes.

1.1.Object Design Trade-Offs

1.1.1Aesthetics and Functionality

For the Univhex project, functionality is more important than aesthetics.

In Univhex, users should use the application with ease. Users should be able to reach any function without a problem or delay. For example, a user can post a photo without unnecessary delay and touches. This may cause some aesthetic design issues but the functionality will be protected.

1.1.2Cost and Quality

For the Univhex project, quality is in the forefront but due to our budget, this has a limit.

Cost is an important feature for this project because we have a limited budget due to the fact that we are students. However, Univhex must be of high queality. That is why we cannot choose or give priority any of these features.

1.1.3Usability and Security

For the Univhex project, usability and security are both significant features. Although they are both important, security is slightly prominent.

Because the application can only reachable for university students, we need to give more resources to the security. Because in case of any information leak or something like that, project and students would be in danger. When we try to provide security, we could lose time and this would reduce usability.

1.2Interface Documentation Guidelines

1.3Engineering Standarts

Univhex uses the UML principles to describe the class interfaces, diagrams, scenarios, use cases, subsystem compositions, and hardware depictions. The UML is a crucial component of the software development process and creating object-oriented software.

1.4Definitions, Acronyms, and Abbreviations

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

Client: The application interacting with the user.

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

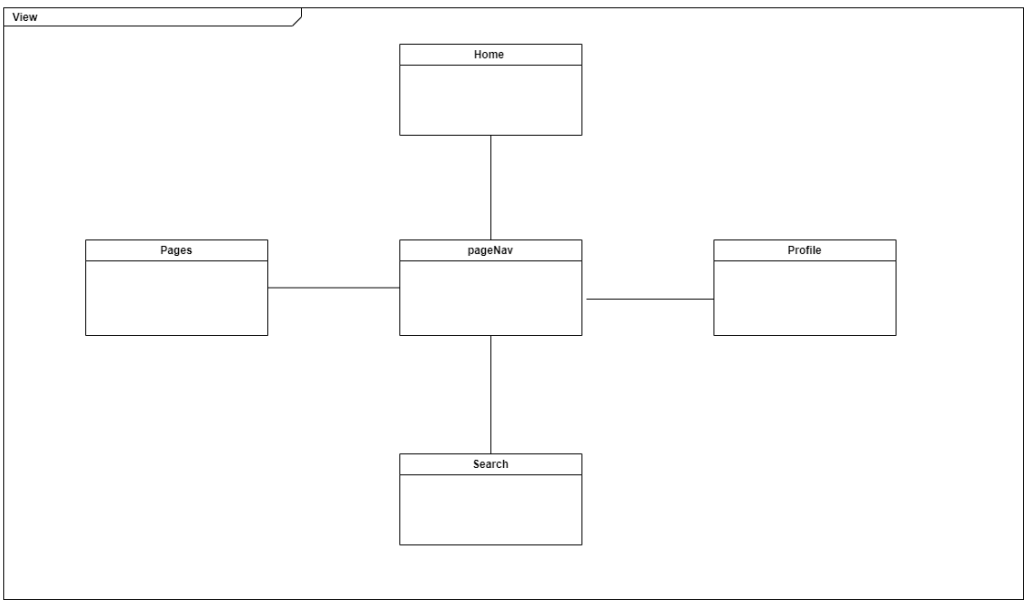
UI: User Interface.

API: Application Programming Interface

2.Packages

Univhex project has 3 different packages: View, Server and Application packages.

2.1View



pageNav: PageNav is the main view for this project. With this, we can navigate through every page.

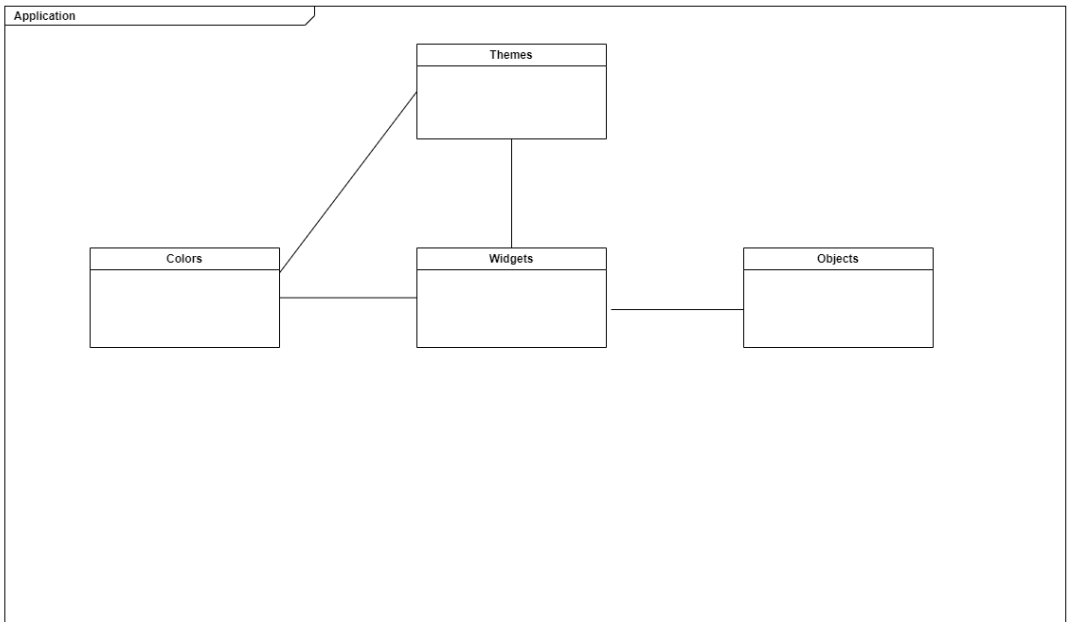
Search: Search page and its information will be held in Search.

Profile: Users info will be held in Profile.

Pages: This contains 3 different pages: Login page, register page and registerContinue page.

Home: Main page of the application.

2.2Application



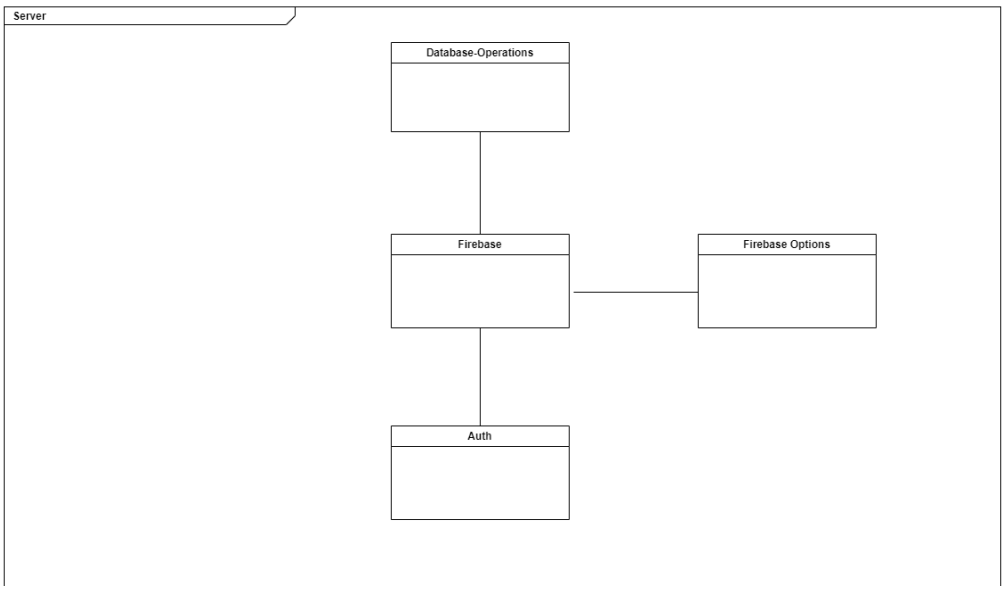
Widgets: This class has every widget that is used in application. These are: buttons, text fields, page forms and helpers.

Colors: Colors that used on the Univhex application is stored in this class. Also, themes class is inherites colors class.

Themes: Page design like login page or home page, is included here.

Objects: Every needed user information stored in this class.

2.3Server



Firebase: Firebase class is where the information will be stored in server.

Firebase Options: This class provides us to change the database with our request.

Auth: This class contains only authentication information. Authentication operation is done in this class instead of Database operations.

Database Operations: In this class, database operations like login or post will be done.

3.Class Interfaces

3.1Pages

3.1.1Login

class Login
Attributes
Methods build**()**

class Register
Attributes const Register({super.key})
Methods build**()**

class RegisterContinue
Attributes Decoration Child backgroundColor appBar body Text _registerFormKey University, major, yearOfStudy
Methods _registerFormKey build()

3.1.2Home

class HomeScreen
Attributes key HomePage
Methods

3.1.3Profile

class ProfileScreen
Attributes Scaffold AppBar SafeArea Text IconButton Center Column addVerticalSpace HexagonWidget AspectRatio Row addHorizontalSpace SizedBox Image.asset

DefaultTabController TabBar TabBarView
Methods build(BuildContext context) addVerticalSpace(BuildContext context, double space) addHorizontalSpace(BuildContext context, double space) addVerticalSpace profileTabBar() createState() _profileTabBarState() initState() build(BuildContext context)

3.1.4Search

class SearchScreen
Attributes
Methods build**()**

3.2Objects

3.2.1AppUser

class AppUser
Attributes name surname email password university fieldOfStudy yearsOfStudy createdAt imgURL hexPoints
Methods AppUser() toString() factory AppUser.fromFirestore() Map<String, dynamic> toFireStore()

3.3Server

3.3.1Firebase

class auth
Attributes
Methods Future<bool?> Register_User(AppUser user) Future<bool> Auth_User(String email, String password)

class database_operations
Attributes
Methods registerAppUserDB(AppUser user) readUserfromDB(String email)

4.Glossary

Flutter: A mobile app development framework that allows for the creation of high-performance, cross-platform applications.

Firebase: A mobile and web application development platform that provides a set of tools and services for building and managing applications, including authentication, database management, and storage.

Cloud Firestore: A flexible, scalable NoSQL document database provided by Firebase.

AppUser: A class that represents a user of the social media app, including attributes such as name, email, password, university, and field of study.

Post: A class that represents a user's post on the social media app, including attributes such as the post content, the date it was created, and the user who created it.

Comment: A class that represents a user's comment on a post, including attributes such as the comment content, the date it was created, and the user who created it.

Like: A class that represents a user's like on a post, including attributes such as the date it was created and the user who created it.

Follow: A class that represents a user's follow relationship with another user, including attributes such as the date the relationship was created and the users involved.

Database Operations: A set of functions and methods for interacting with the Firebase Cloud Firestore database, including functions for creating, reading, updating, and deleting user data, posts, comments, likes, and follows.

Authentication: The process of verifying a user's identity, typically through a login system that requires a valid email and password.

Navigation: The process of moving between different screens or views within the app, typically facilitated through a navigation bar or drawer.

State Management: The process of managing and updating the state of the app's user interface, typically through the use of stateful widgets and provider patterns.

Widgets: The building blocks of the Flutter UI, including basic widgets such as text, images, and buttons, as well as more complex widgets such as lists, forms, and animations.

API: An application programming interface that allows for the exchange of data between different software applications, typically through a set of rules and protocols. In the context of a social media app, an API might be used to connect the app to external services such as Google Maps or Twitter.

5.References

- Flutter documentation: https://flutter.dev/docs
- Firebase documentation: https://firebase.google.com/docs
- Cloud Firestore documentation: https://firebase.google.com/docs/firestore
- FlutterFire documentation: https://firebase.flutter.dev/
- "Flutter & Firebase: Build a Complete Social Network App for iOS & Android" Udemy course by Phil Jay: https://www.udemy.com/course/flutter-firebase-build-a-complete-social-network-app-for-ios-android/
- "Flutter Social App UI Kit" by Flutter Templates: https://flutter-templates.com/flutter-social-app-ui-kit/
- "Flutter Social Media App UI Kit" by Enappd: https://enappd.com/flutter-social-media-app-ui-kit/overview/6
- "Flutter Instagram Clone" by Mitesh Shah: https://github.com/mitesh77/flutter_instagram_clone