Bilkent University

Department of Computer Engineering

Senior Design Project

QUEX

Low-Level Design Report

Project Members:

Barış Ardıç 21401578

Emir Acımış 21201233

Umutcan Aşutlu 21301093

Mert Kara 21400976

Atakan Özdemir 21301134

Supervisor: Çiğdem Gündüz Demir

Jury Members: Fazlı Can, Hamdi Dibeklioğlu

Table of Contents

1.Introduction

The main purpose of Quex is to find its users trustworthy individuals when they need help in their daily routine. The program provides help about variety of topics like computer hardware or software, daily decision-making, travelling, entertainment or dining. Even the simplest problems we encountered in our daily routine may take lots of time if we cannot get assistance from a person who is experienced with our problem. This is why we focus on “expert finding”, the program searches for nearest expert about a problem and match user with expert so that they can talk and share knowledge about the program via online chat. The purpose of using location based expert matching is, if the users trust each other, they can meet to solve the problem. If program fails to find an expert around the user or the expert cannot solve the problem of the other user, program pushes the problem directly to main dashboard under a certain category. The problems under the dashboard can be seen by every user of the system. So, if an expert desires to help someone, they can directly chat with the user. The user which pushed the problem to the QUEX can also write down “unsuccessful solutions” under the problem which is in the dashboard, so other users can see and suggest a different solution. After the chat between user and the expert closes, user can give “+” rating to the expert. This rating will be used for making other users comfortable about the expert.

1.1 Object design trade-offs

Functionality vs Ease of Use

Quex needs to have a wide range of active users, because a crowded community provides a better cooperation among users. Especially, due to the fact that Quex focuses on expert finding, it needs lots of users to match them with each other. So that ease of use is

one of the most important feature of Quex. Some features are restricted to eliminate complex interface risks.

Traditional vs Innovative

Innovative interfaces bring risks with it since users generally are not familiar with new interfaces and designs in applications. There is always a risk that users may find interface too difficult to adapt. Because of that, we decided to adopt a traditional layout to make users quickly learn about the program. It also eases the use of the program.

Performance vs Cost

Quex should be an “agile” program which means it needs fast response time when it is searching and matching. Many users may be search for an expert simultaneously, so they need to reach a solution in a short time. So that performance is a very serious issue and it is definitely needed, it is clear that we choose performance over cost.

Security vs Cost

Because users will talk with many of other users in the program, it is important to provide that their personal information like open address and location out of limits for the others. Also censorship of unwanted or inappropriate chat materials like swear words, nudity and bank account numbers is an important matter. Security of Quex should be perfect to reach this goal, so we go with security over cost.

1.2 Interface documentation guidelines

Name of the Class	
Description	Description of the class
Attribute	Attributes of the class
Operation	Operations of the class (void if not mentioned)

1.3 Engineering standards

The low level design report is written in IEEE standard engineering writing form which is a widely accepted by engineers. The UML which is very common in industry is used for graphical representation of the system with its components and classes.

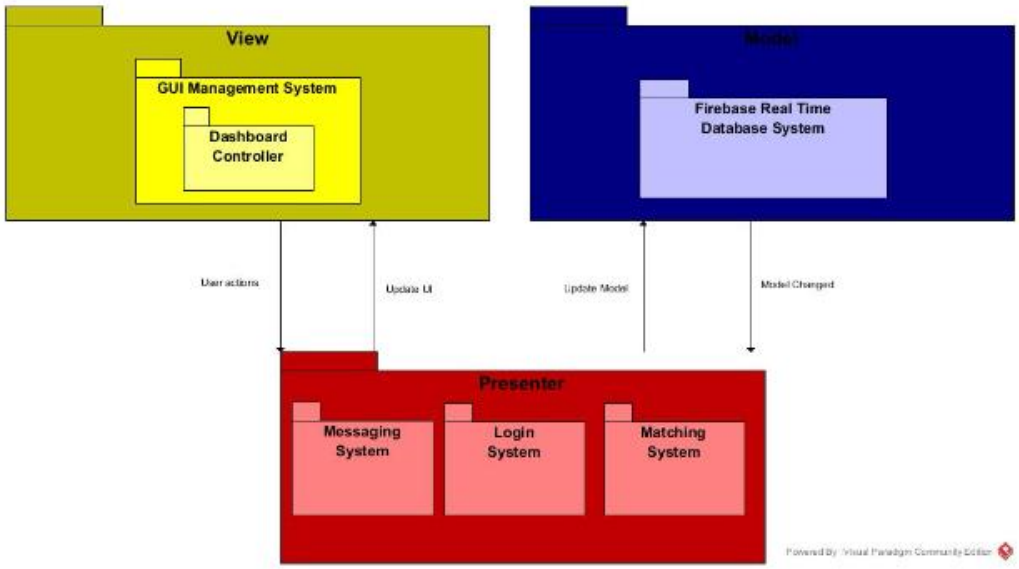
1.4 Definitions, acronyms, and abbreviations

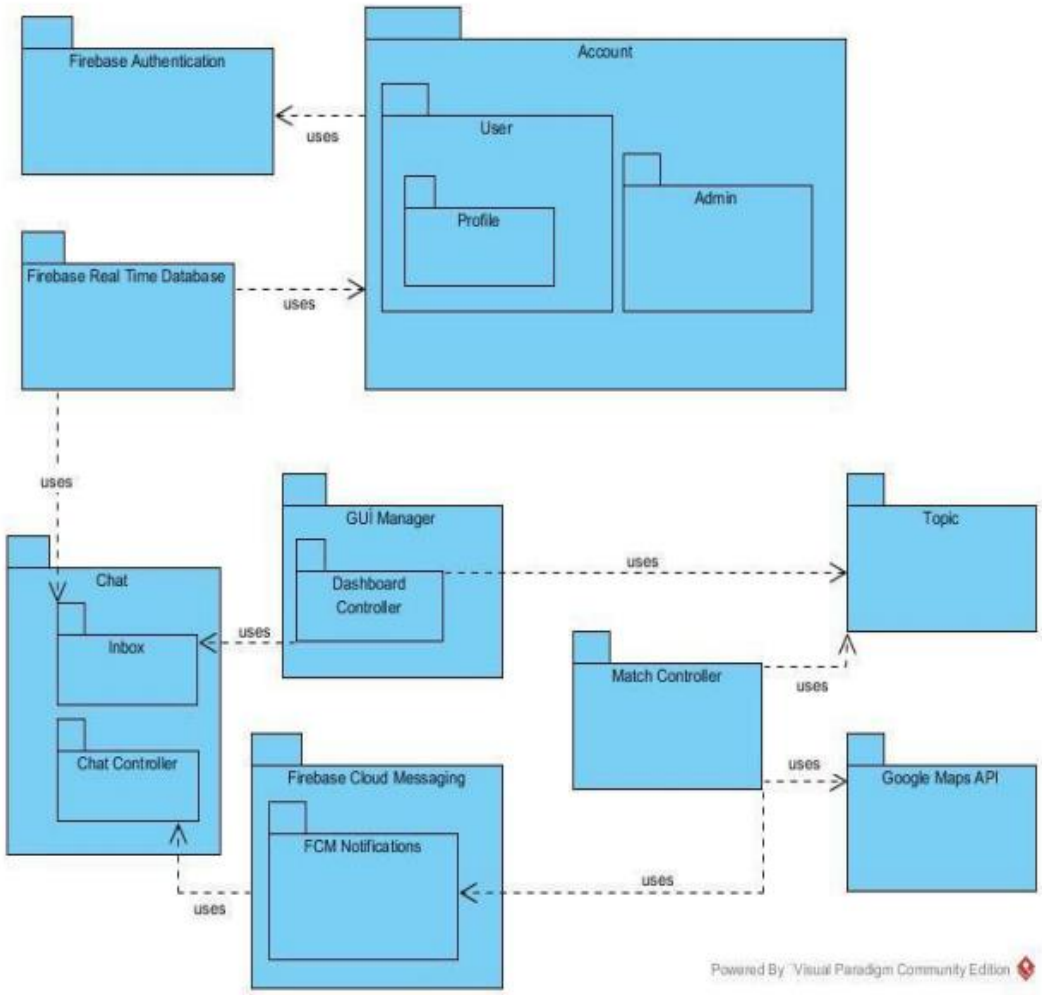
UML: Unified Modeling Language

IEEE: The Institute of Electrical and Electronics Engineers

2. Packages

Quex' system makes use of MVP architecture. The MVP architectural style is selected for QUEX application because it has integrated XML modules in it. These modules do not have any logic or decision making in them. They are used for style and user interface design. This allows for XML components to be seen as views. Our system is decomposed into 4 packages in compliance with the following diagrams as ViewPackage, ControllerPackage, ModelPackage and DataPackage.





In relation to figure-2, for example, the dataPackage includes the Firebase services. Topic, User, Inbox and Chat systems are included in the modelPackage, ChatController and MatchController are included in the controllerPackage and dashBoardController is in viewPackage.

2.1Client

Client subsystem is the user part of the program. It is also responsible for presentation of the data it collects from server to user, managing the operations of user, and send necessary notifications.

2.1.1 ViewPackage

View package consists of view objects which are the visible part of the program to the user. It provides user with the interface elements. Main purpose of these are to display the data from the model objects and create the environment for management of the data.

2.1.2 ControllerPackage

ControllerPackage is responsible for the bridging of model and view classes. It includes the classes for subsystems such as messaging and matching. It manages the connection to Firebase API. It manages and sends queries to API, and then, receives and draws out the results via connection to viewPackage. It manages the connection with the network.

2.1.3 ModelPackage

ModelPackage classes have relationships with View and Controller classes and constructs the mains template of objects of Quex, such as User, Topic and Message classes.

2.2 Server

2.2.1 DataPackage

This package provides the interface to handle the connection, the API requests and responses between the server and its clients while providing the interface to make queries to the database to find topics or experts. It is responsible for simple database transactions, insertions, updates or deletes. In addition, it includes the backend logic of matching, searching and messaging subsystems.

3. Class Interfaces

LoginOrRegister Activity	
Description	The first activity of the application which direct user to Login activity or register activity. The Google login and Facebook login is also included in this activity.
Attribute	-mLogin : Button -mRegister : Button -signInButton : Button -Rc_sign_in : int -mAuth : FirebaseAuth -mAuthListener:FirebaseAuth.AuthStateListener -mGoogleSignInClient : GoogleSignInClient -TAG : String
Operation:	+onCreate(Bundle savedInstanceState) +onStart() +onActivityResult(int requestCode, int +resultCode, Intent data) +firebaseAuthWithGoogle(GoogleSignInAccount account) +signIn()

Login Activity	
Description	This class is for handling login activity for the user. The basic inputs are email and password given by the user.
Attribute	-mLogin:Button -mEmail:EditText -mPassword:EditText -mAuth:FirebaseAuth -mAuthStateListener:FirebaseAuth.AuthStateListener
Operation	+onCreate(Bundle savedInstanceState) +onClick(View v) +onComplete(@NonNull Task<AuthResult> task) +onStart() +onStop()

Register Activity	
Description	This class is handling register activity for the user to create an account for using the application.
Attribute	-mRegister : Button -mEmail : EditText -mPassword : EditText -mAuth : FirebaseAuth -mAuth : FirebaseAuth.AuthStateListener
Operation	+onCreate(Bundle savedInstanceState) +onStart() +onStop()

User Class	
Description	The instances of this class holds user information referring to a unique user session.
Attribute	-Name: String -Surname : String -Email : String -Bio : String -UserId : String
Operation	+getUserID() +onCreate(Bundle savedInstanceState) +onStart() +onStop()

Dashboard Activity	
Description	This class is showing topics on the system that if users did not find a match for topics they write. It is added on the dashboard after match activity is done.
Attribute	-topic:Topic[]
Operation	+displayTopic() +sortTopic(Category c) +filterTopic(Category c) +onCreate(Bundle savedInstanceState) +onStart() +onStop()

Topic Class	
Description	The created topic is added to the system with this class.
Attribute	-cat:Category -TopicDesc:EditText -solved:boolean
Operation	+sentDashboard() +sentMatch() +onCreate(BundlesavedInstanceState)

Profile Activity	
Description	Saves user information for app functions, also displays the information back to the user as a page.
Attribute	-mDone:Button -mJob:EditText -mBio:EditText -mPhone:EditText -mAuth: FirebaseAuth -mAuthListener:FirebaseAuth.AuthStateListener -mDatabase: DatabaseReference
Operation	+onCreate(Bundle savedInstanceState) +onStart() +onStop() +onClick(View v) +onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth)

Match Activity	
Description	This module has the algorithm and intents of the matching functionality
Attribute	-Questioneer:User -top:Topic
Operation	+getLocation() +matchMaxDistance:float +findMatch(Questioneer,top) +findMatchL(Questioneer,top) +onCreate(Bundle savedInstanceState) +onStart() +onStop()

Inbox Activity	
Description	This class is for handling the inbox for the user. It shows the past and present messages that the user sent out and received.
Attribute	-chats:Chat[]
Operation	+displayChat(Chat c) +deleteChat(Chat c) +onCreate(Bundle savedInstanceState) +onStart() +onStop()

Chat Class	
Description	This class is handling the users do the messaging with the other users. Users get to send/receive messages with this class.
Attribute	-Questioneer:User -Expert:User -top:Topic -mes:String[] -solved:boolean
Operation	+sentMes(Sting message) +MesViewed() +sentLoc() +TopicSolved() +onCreate(Bundle savedInstanceState) +onStart() +onStop()

4. Glossary

Android: Mobile phone operating system

QUEX: Name of the application

UML: Undefined Modeling Language

GPS: The global positioning system

MVP: Model-View-Presenter

5. References

[1] Firebase. (2017). Firebase. [online] Available at: https://firebase.google.com/ [Accessed 11 Feb. 2018].

[2] Google Developers. (2017). The Google Maps Geolocation API, Google Maps Geolocation API | Google Developers. [online] Available at: https://developers.google.com/maps/documentation/geolocation/intro [11 Feb. 2018].

[3] Developer.android.com. (2017). Android Developers. [online] Available at: https://developer.android.com/index.html [Accessed 11 Feb. 2018].

[4] Amazon Web Services, Inc. (2017). What is NoSQL? – Amazon Web Services (AWS). [online] Available at: https://aws.amazon.com/nosql/ [Accessed 11 Feb. 2018].