Software Design Description

IZTECH
Graduation Application

Project Members

250201007 - Zekihan AZMAN
250201036 - Kaan ALGAN
250201035 - Efe Can CEVHER
240201019 - Baran ATEŞ
240201022 - Kemal Selçuk KAPLAN

Version	Date	Description of Changes
1.0	25/April/2020	Initial viewpoint designs
1.1	29/April/2020	First draft, containing a part of each viewpoint.
1.2	7/May/2020	Completed Logical and Interaction Viewpoints
2.0	10/May/2020	Added missing document elements. Finalized for release.
2.1	16/May/2020	Corrections made according to the review report. ApplicationId now used as the key of Applications (before it was matching userId). Department identifier now held as integer instead of string. Necessary changes made in data model diagram and logical viewpoints accordingly. Improved interface viewpoints. (versions, methods, detailed explanation of services)
3.0	18/May/2020	Improved design rationale. (added some unmentioned viewpoints, more explanation on why we preferred this design) Improved “Identified Stakeholders and Design Concerns” section by explaining each stakeholder.

Table of Contents

1. Introduction

1.1 Purpose

Purpose of this document is to serve as a guideline throughout the development phase of this project for other developers. This document provides necessary information about the design of the Graduate Program Application whose requirements and functionalities were summarized in Software Requirements Specifications (SRS) report. Aim is to provide a clear understanding of the design so it can be developed easily later by any developer reading this document.

1.2 Scope

This document will contain the general definition and features of the project, the identified stakeholders and design concerns, the overall system architecture and data architecture by providing different viewpoints and the system interface. With the help of UML diagrams and MVC design of the system and subsystems/modules will be explained visually in order to help the developers to understand all information stated in this document correctly and easily.

1.3 Context

Software aims to provide a service where all applicants', department users' and grad school officials' can interact with each other by means of application process, safely and securely.

All functionalities of all types of users are well-defined and separated with the help of assigning roles to users to perform particular operations. (e.g. only computer engineering department users can set or cancel interviews of computer engineer department applicants)

Applicants' documents and personal information are stored securely, accessed by the authorized personnel, that is, only the corresponding department users. HTML/CSS Javascript and Firebase are the main components that are going to be used in developing the program.

The product will be designed with a modular approach with the help of MVC design pattern because it will make it easy to extend the software functionality later in the future if needed by the customers.

The document is prepared by adhering to the IEEE standards (IEEE Std 1016 – 2009).

1.4 Summary

Logical viewpoint gives an idea about the structure of the program, representative of the MVC design model, three basic components for each 'action'. Under each logical viewpoint is there a definition of shown methods.

Interaction viewpoint gives an idea about the flow of the program, shows the methods and interaction of components with each other, and a detailed representation of how the program works. Each interaction viewpoint is given a brief description below it, explaining the flow of that use case.

Information viewpoint gives an idea about how the required data is handled(stored). A definition of how the program uses the database is given below the diagram.

Interface viewpoint gives an idea about external dependency on other interfaces, also a brief description of how the data flows to/from different services to the product is given.

2. References

[1] 1016-2009 - IEEE Standard for Information Technology--Systems Design--Software Design Descriptions. (2009, July 20). Retrieved May 10, 2020, from https://standards.ieee.org/standard/1016-2009.html

[2] Unified Modeling Language. (2017, December). Retrieved May 10, 2020, from https://www.omg.org/spec/UML/

[3] ALGAN, K., CEVHER, E., AZMAN, Z., ATEŞ, B., & KAPLAN, K. (April 14 2020). Graduate Program Application Software Requirements Specification Document.

[4] What is event-driven architecture? (n.d.). Retrieved May 10, 2020, from https://www.redhat.com/en/topics/integration/what-is-event-driven-architecture

[5] Bhattacharjee, A., & Sap. (2014, May 08). NoSQL vs SQL – Which is a Better Option? Retrieved May 10, 2020, from https://blog.udemy.com/nosql-vs-sql-2/?utm_source=adwords

[6] Facebook Login for the Web with the JavaScript SDK. (n.d.). Retrieved May 16, 2020, from https://developers.facebook.com/docs/facebook-login/web

[7] Integrating Google Sign-In into your web app. (n.d.). Retrieved May 16, 2020, from https://developers.google.com/identity/sign-in/web/sign-in

[8] Cascading Style Sheets, HTML, NoSQL, MVC, UML. (2020, April 28). Retrieved May 10, 2020, from https://en.wikipedia.org/wiki/

3. Glossary

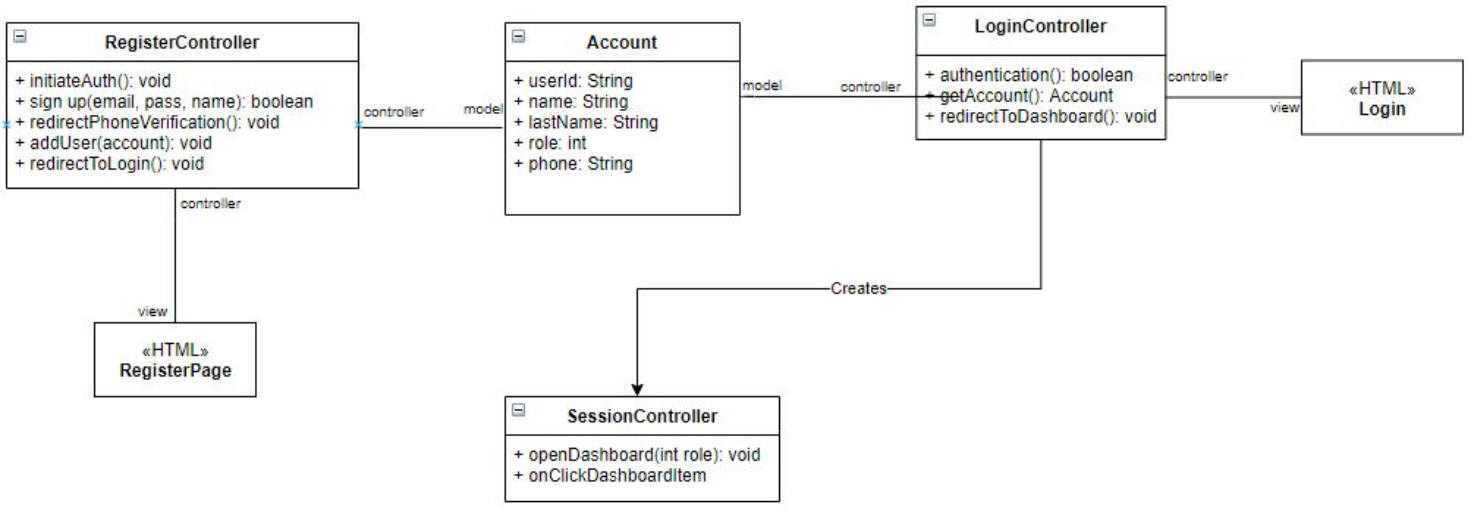
Term	Definition
NoSQL	A NoSQL is a type of database that provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.
UML	The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.
MVC	Model–view–controller (usually known as MVC) is a software design pattern commonly used for developing user interfaces which divides the related program logic into three interconnected elements.
SQL	Structured Query Language) is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS),
HTML	Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser.
CSS	Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML.
SDD	A software design description is a written description of a software product that a software designer writes in order to give a software development team overall guidance to the architecture of the software project.

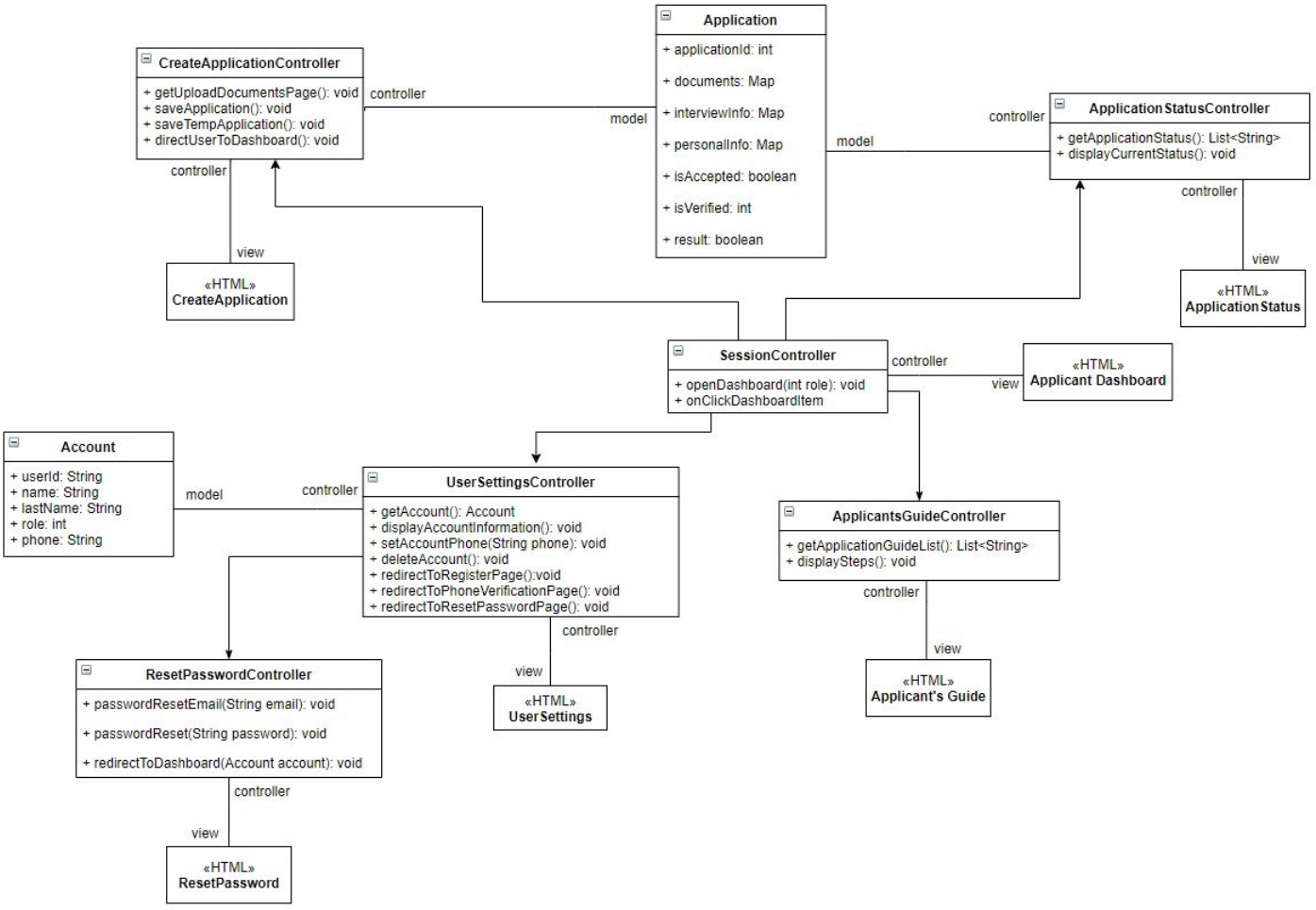
4.1 Identified Stakeholders and Design Concerns

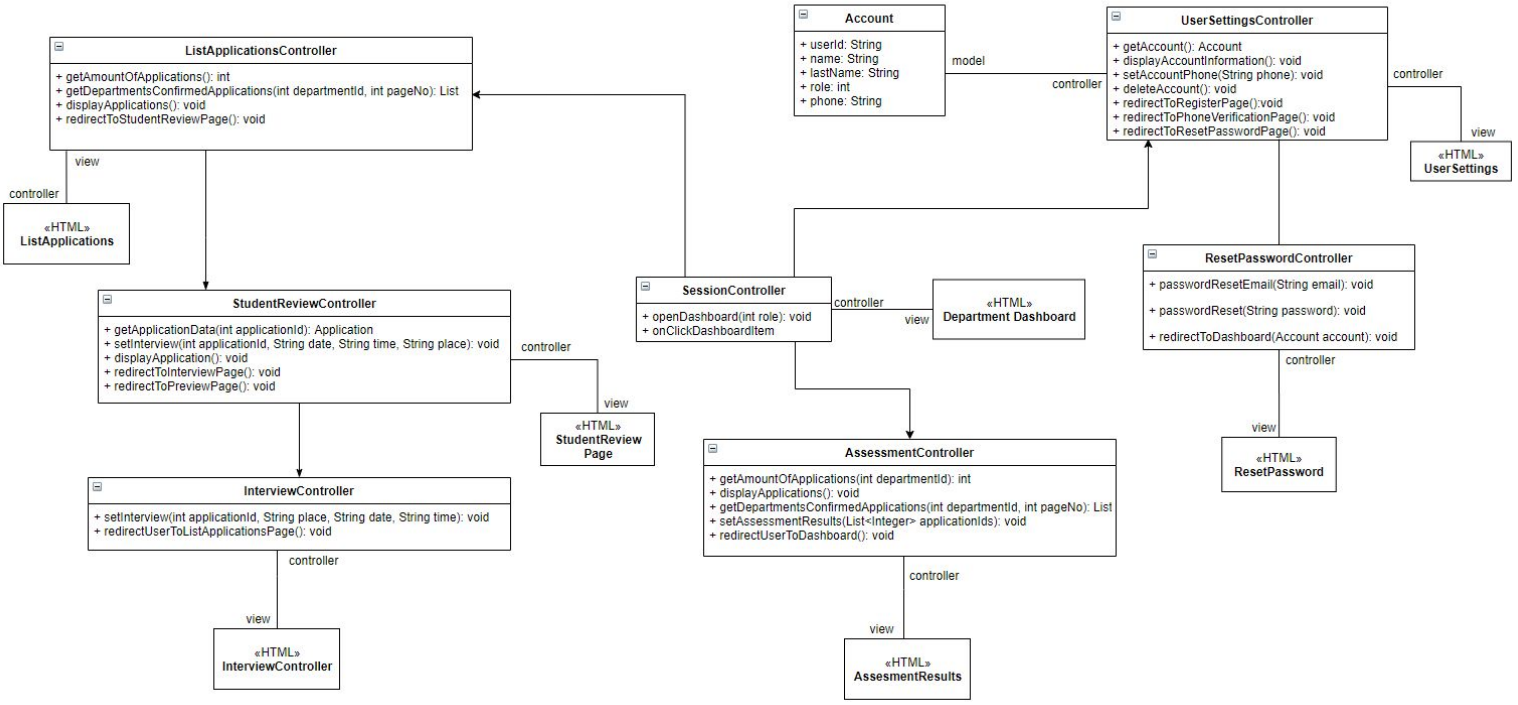
Stakeholders include developers, maintainers, testers and end users of “Graduate Program Application”. Since it will be a web application, usage should be easy, interfaces should be practical and appealing.

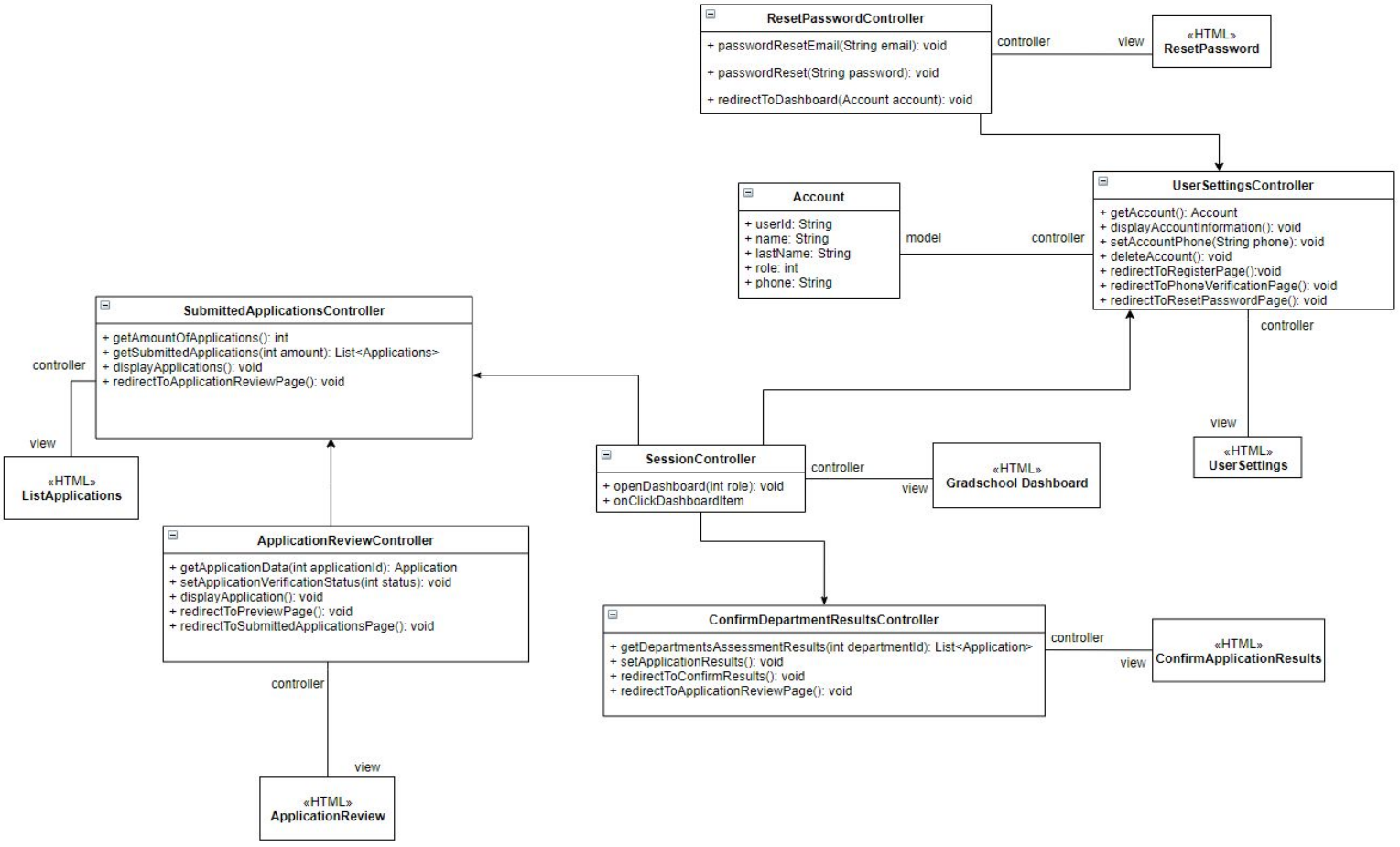
- Testers will work with the interaction viewpoints to write programs to test functionality of each use case, and trace bugs.
- Developers will mainly use logical viewpoint to understand the logic behind the system.
- Maintainers will mainly use information viewpoint and interface viewpoint to maintain and upgrade necessary interfaces accordingly.

4.2 Logical viewpoints

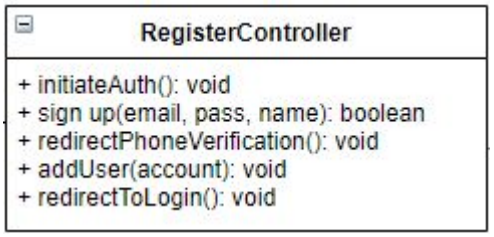





ListApplicationsController
+ getAmountOfApplications(): int
+ getDepartmentsConfirmedApplications(int departmentId, int pageNo): List
+ displayApplications(): void
+ redirectToStudentReviewPage(): void
view
controller
«HTML»
ListApplications
StudentReviewController
+ getApplicationData(int applicationId): Application
+ setInterview(int applicationId, String date, String time, String place): void
+ displayApplication(): void
+ redirectToInterviewPage(): void
+ redirectToPreviewPage(): void
controller
view
«HTML»
StudentReview
Page
InterviewController
+ setInterview(int applicationId, String place, String date, String time): void
+ redirectUserToListApplicationsPage(): void
controller
view
«HTML»
InterviewController
Account
+ userId: String
+ name: String
+ lastName: String
+ role: int
+ phone: String
model
controller
UserSettingsController
+ getAccount(): Account
+ displayAccountInformation(): void
+ setAccountPhone(String phone): void
+ deleteAccount(): void
+ redirectToRegisterPage():void
+ redirectToPhoneVerificationPage(): void
+ redirectToResetPasswordPage(): void
controller
view
«HTML»
UserSettings
SessionController
+ openDashboard(int role): void
+ onClickDashboardItem
controller
view
«HTML»
Department Dashboard
ResetPasswordController
+ passwordResetEmail(String email): void
+ passwordReset(String password): void
+ redirectToDashboard(Account account): void
controller
view
«HTML»
ResetPassword
AssessmentController
+ getAmountOfApplications(int departmentId): int
+ displayApplications(): void
+ getDepartmentsConfirmedApplications(int departmentId, int pageNo): List
+ setAssessmentResults(List<Integer> applicationIds): void
+ redirectUserToDashboard(): void
controller
view
«HTML»
AssesmentResults

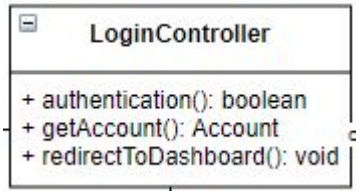

ResetPasswordController
+ passwordResetEmail(String email): void
+ passwordReset(String password): void
+ redirectToDashboard(Account account): void
controller
view
«HTML»
ResetPassword
Account
+ userId: String
+ name: String
+ lastName: String
+ role: int
+ phone: String
model
controller
UserSettingsController
+ getAccount(): Account
+ displayAccountInformation(): void
+ setAccountPhone(String phone): void
+ deleteAccount(): void
+ redirectToRegisterPage():void
+ redirectToPhoneVerificationPage(): void
+ redirectToResetPasswordPage(): void
controller
view
«HTML»
UserSettings
SubmittedApplicationsController
+ getAmountOfApplications(): int
+ getSubmittedApplications(int amount): List<Applications>
+ displayApplications(): void
+ redirectToApplicationReviewPage(): void
controller
view
«HTML»
ListApplications
SessionController
+ openDashboard(int role): void
+ onClickDashboardItem
controller
view
«HTML»
Gradschool Dashboard
ApplicationReviewController
+ getApplicationData(int applicationId): Application
+ setApplicationVerificationStatus(int status): void
+ displayApplication(): void
+ redirectToPreviewPage(): void
+ redirectToSubmittedApplicationsPage(): void
controller
view
«HTML»
ApplicationReview
ConfirmDepartmentResultsController
+ getDepartmentsAssessmentResults(int departmentId): List<Application>
+ setApplicationResults(): void
+ redirectToConfirmResults(): void
+ redirectToApplicationReviewPage(): void
controller
view
«HTML»
ConfirmApplicationResults

Register Controller



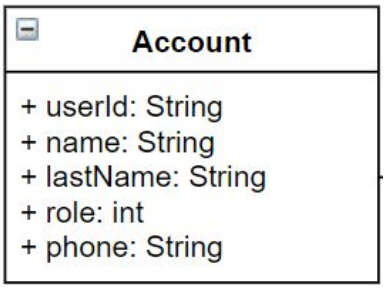
Method	Description
initiateAuth	returns void, starts the authentication process by starting firebase auth library
signUp	takes three string respectively email password and name, returns boolean indicating success of operation, creates an account.
redirectPhoneVerification	returns void, redirects user to phone verification.
addUser	takes account as parameter, returns void, creates and populates respective fields in the database.
redirectToLogin	returns void, redirects user to the login page.

4.2.1 Login Controller



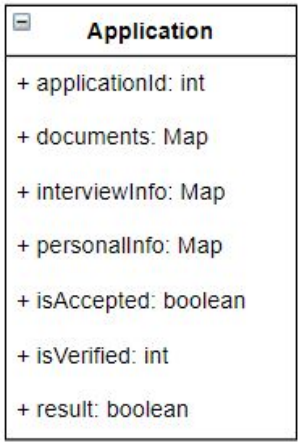
Method	Description
authentication	returns boolean indicates success of operation, authenticate user credentials.
getAccount	returns account, creates an account of the user that has logged in.
redirectToDashboard	returns void, redirects user to the dashboard page.

4.2.2 Account



Attribute	Description
userId	String, a unique string for user.
name	String, name of the accounts owner
lastName	String, last name of the accounts owner
role	int, role of the user which can be applicant, department, grad school. Each role gives different abilities to user.
phone	String, phone number of the accounts owner

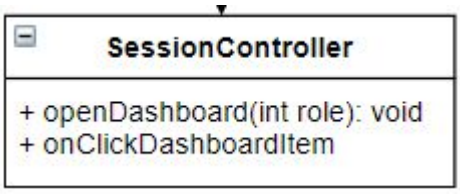
4.2.3 Application



Attribute	Description
applicationId	An int value, unique identifier of an application.
documents	Map which contains the necessary documents of the corresponding applicant.
personalInfo	Map which contains the personal information of corresponding applicant such as Name,Last Name,Phone Number…

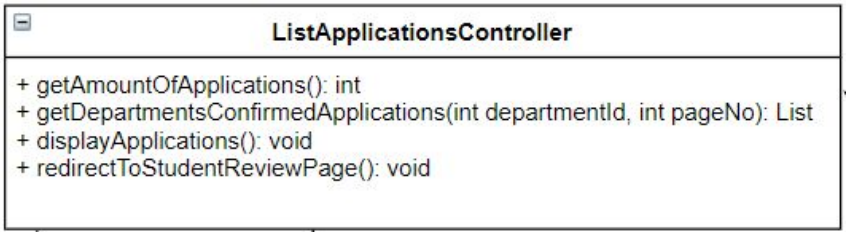
interviewInfo	Map which contains the interview information (date,place...) of the corresponding applicant.
isAccepted	A boolean value that indicates whether the application is accepted by the department or not.
isVerified	An int value indicates whether the application is verified by the grad school.
result	A boolean value that indicates whether the applicant to whom the application belongs, is accepted to the school and announced by the grad school.

4.2.4 Session Controller



Method	Description
openDashboard	takes int which is role of user, returns void, opens different dashboards for each role.
onClickDashboardItem	returns void, redirects user appropriate page for that menu item.

4.2.5 List Applications Controller



Method	Description
getAmountOfApplications	returns the number of applications that is present in the database to be able to divide up the applications into separate pages
getDepartmentsConfirmed Applications	takes page number as one of its parameters to get the particular range of applications from the database (e.g. if the pageNo is 1 then a list of the applications from id=20 to id=40 that belongs to the department with the given departmentId argument will be returned (if 20 applications will be displayed per page).
redirectToStudentReview Page	redirects the department user to the selected applicant's review page.

displayApplications	Converts each application data into a HTML element and adds them into the program tree to display.

4.2.6 User Settings Controller

UserSettingsController

+ getAccount(): Account
+ displayAccountInformation(): void
+ setAccountPhone(String phone): void
+ deleteAccount(): void
+ redirectToRegisterPage():void
+ redirectToPhoneVerificationPage(): void
+ redirectToResetPasswordPage(): void

Method	Description
getAccount	returns Account gets account information from the database.
displayAccountInformation	displays account information on the account settings page.
setAccountPhone	takes phone number string as parameter returns void sets new phone string on database.
deleteAccount	returns void deletes the account from the database.
redirectToRegisterPage	redirects user to register page after account deletion.
redirectToPhoneVerificationPage	redirects user to the phone verification page to verify the new phone number before setting it.
redirectToResetPasswordPage	redirects user to reset password page to start password reset process

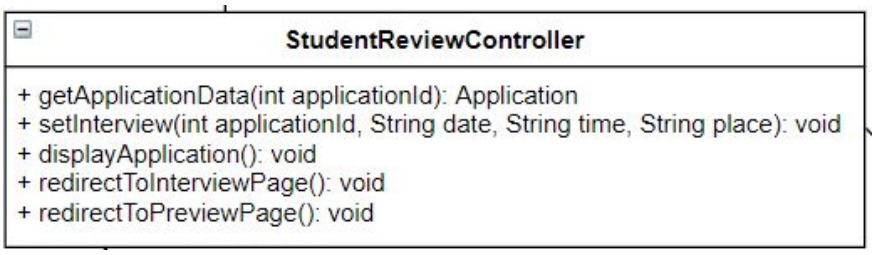
4.2.7 Reset Password Controller

ResetPasswordController

+ passwordResetEmail(String email): void
+ passwordReset(String password): void
+ redirectToDashboard(Account account): void

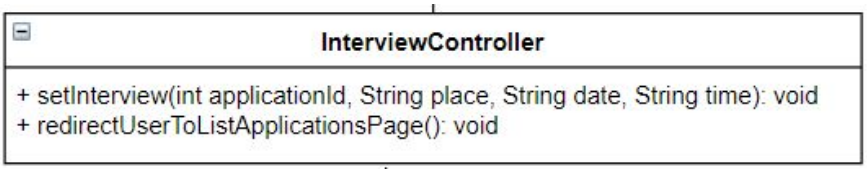
Method	Description
passwordResetEmail	takes email string as parameter returns void sends verification email over firebase authentication to the user to start password reset process
passwordReset	takes password string as parameter returns void sets new password on firebase authentication system
redirectToDashboard	takes account object as parameter returns void, redirects user to his/her dashboard page.

4.2.8 Student Review Controller



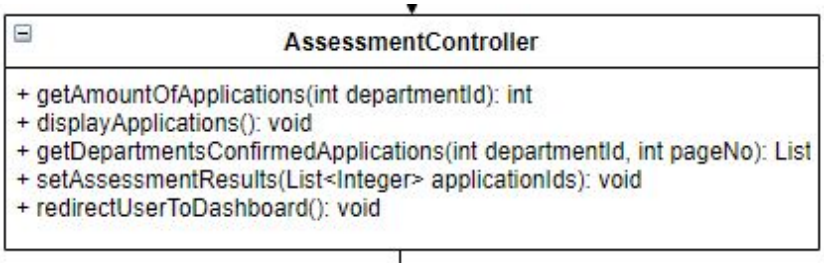
Method	Description
getApplicationData	returns the application with the given id.
displayApplication	Converts the application data into a HTML element and adds it into the program tree to display.
redirectToInterviewPage	Redirects the user to the interview page whenever the ‘Set an interview’ button is clicked.
redirectToPreviewPage	Opens up a new tab in which content of the selected document will be shown, redirects the user there.

4.2.9 Interview Controller



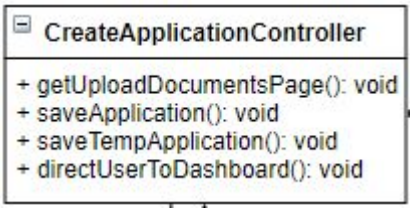
Method	Description
setInterview()	sets an interview for the specified time and place for the application with the given id. returns void
redirectUserToListApplicationsPage()	redirects the user back to the list applications page whenever an interview is set. returns void

4.2.10 Assessment Controller



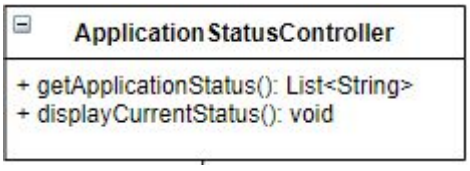
Method	Description
getAmountOfApplications	returns the total number of applications that belong to the specified department from the database.
getDepartmentsConfirmedApplications	returns the confirmed applications by department id and page number.
displayApplications	displays applications in a list.
setAssessmentResults	takes List<Integer> applicationIds sets the assessment results of the applications, selected by the department, in the database.
redirectToUserToDashboard	Redirects the user to the his/her Dashboard.

4.2.11 Create Application Controller



Method	Description
getUploadDocumentsPage()	Redirects the user to the upload document page after the user clicks the next button.
saveApplication()	saves the completed application to the database.
saveTempApplication()	saves the incomplete application ,containing only personal information of the user, to the database.
directUserToDashboard()	Redirects the user to the his/her Dashboard.

4.2.12 Application Status Controller

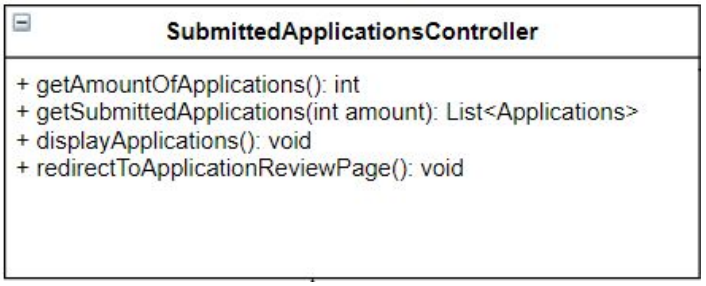


Method	Description
getApplicationStatus()	returns the information(status) about the application as a list of strings from the database.
displayCurrentStatus()	Displays current information(status) of the user's application. Information includes whether the application is confirmed or not, also the place and time of the interview if the application has been confirmed.

4.2.13 Applicants Guide Controller

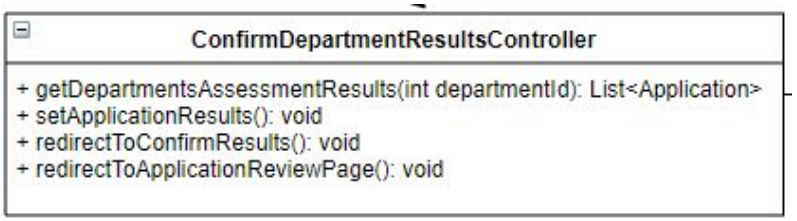
getApplicationGuideList	returns List<String> gets application guide step list from database
displaySteps	display steps on applicant guide page

4.2.14 Submitted Applications Controller



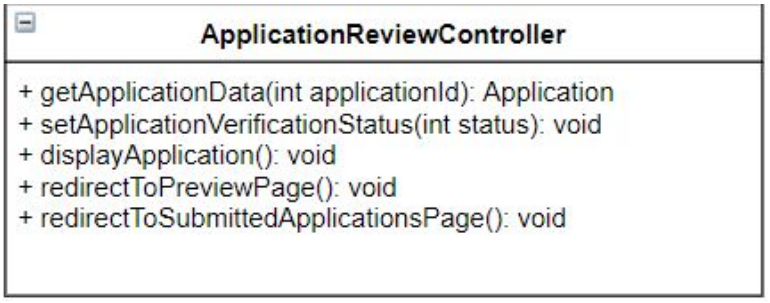
Method	Description
getAmountOfApplications()	returns the total amount of submitted applications from database
getSubmittedApplications(int amount)	returns the list of submitted applications from database
displayApplication()	Converts the list of applications data into HTML elements and adds them into the program tree to display
redirectToApplicationReviewPage()	Redirects the user to the Application Review Page

4.2.15 Confirm Department Results Controller



Method	Description
getDepartmentsAssessmentResults(int departmentId)	returns the list of assessment results of the given department from database
setApplicationResults()	it sets the chosen applications' " result" field true in the database
redirectToConfirmResults()	Redirects the user to the Confirm Results Page
redirectToApplicationReviewPage()	Redirects the user to the Application Review Page

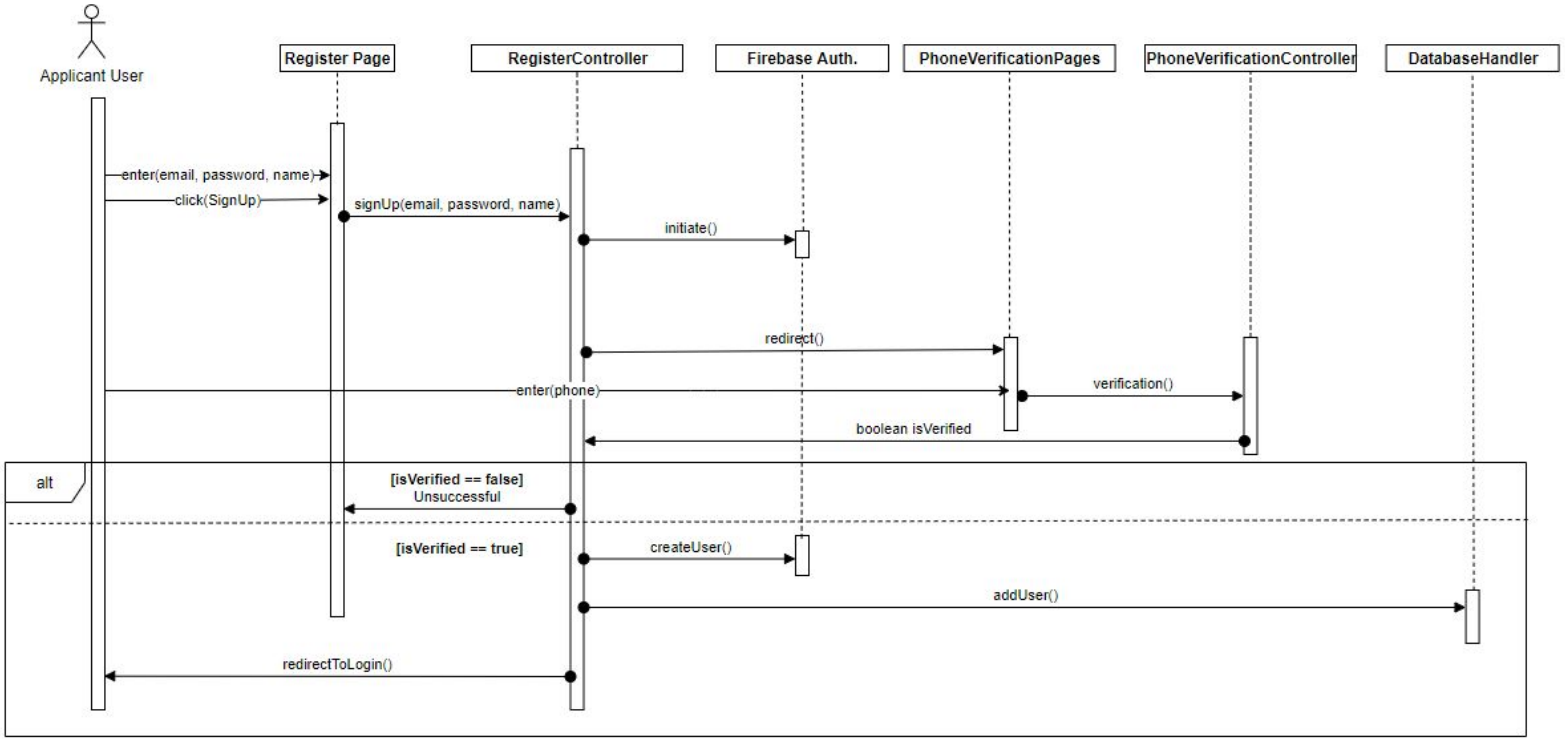
4.2.16 Application Review Controller



Method	Description
getApplicationData()	returns the application with the given id from database
setApplicationVerificationStatus(int status)	it sets the application's "isVerified" field with integer status in the database
displayApplication()	Converts the application data into a HTML element and adds it into the program tree to display.
redirectToPreviewPage()	Opens up a new tab in which the content of the selected document will be shown, redirects the user there
redirectToSubmittedApplicationsPage()	.Redirects the user to the Submitted Applications Page

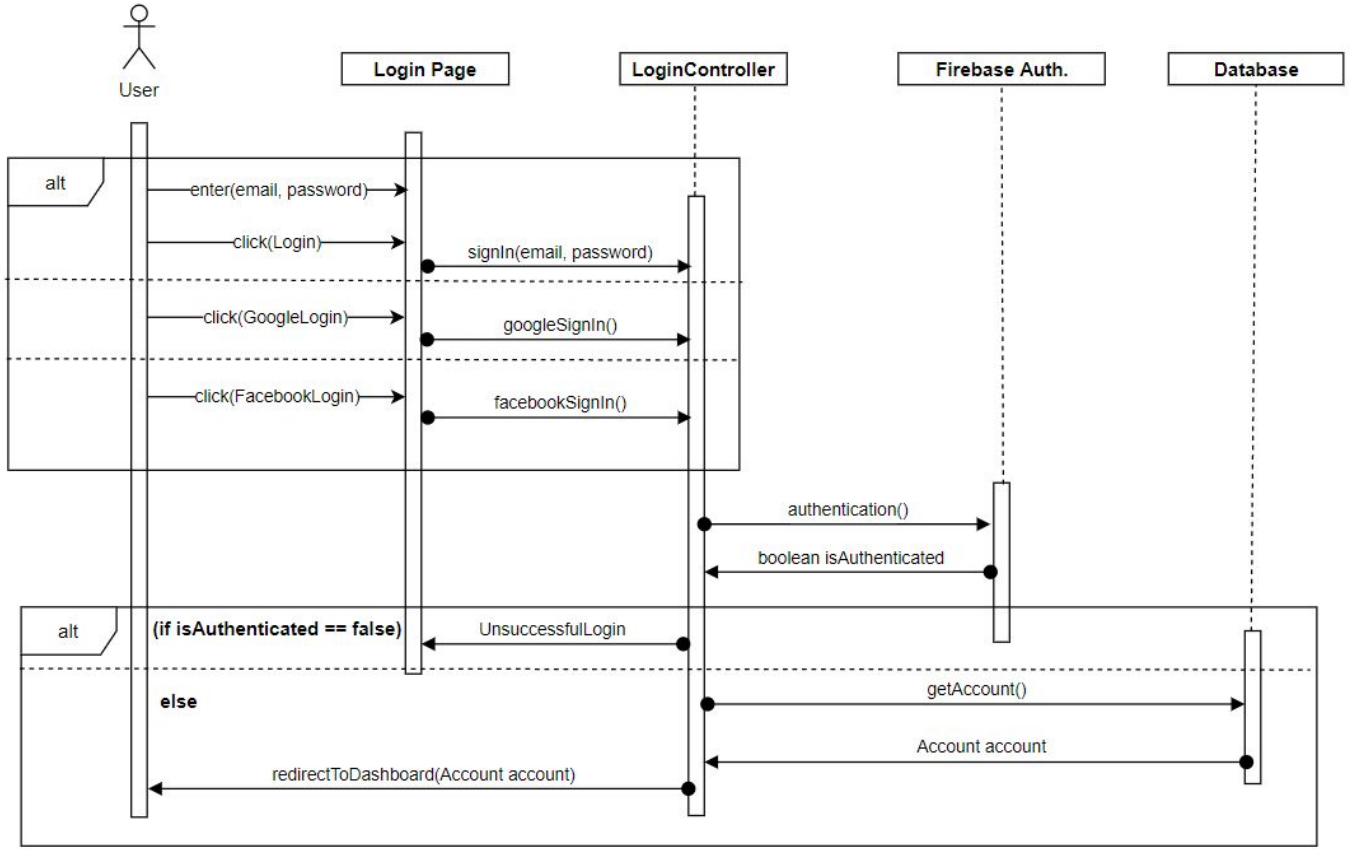
4.3 Interaction viewpoints

4.3.1 Register Interaction Viewpoint



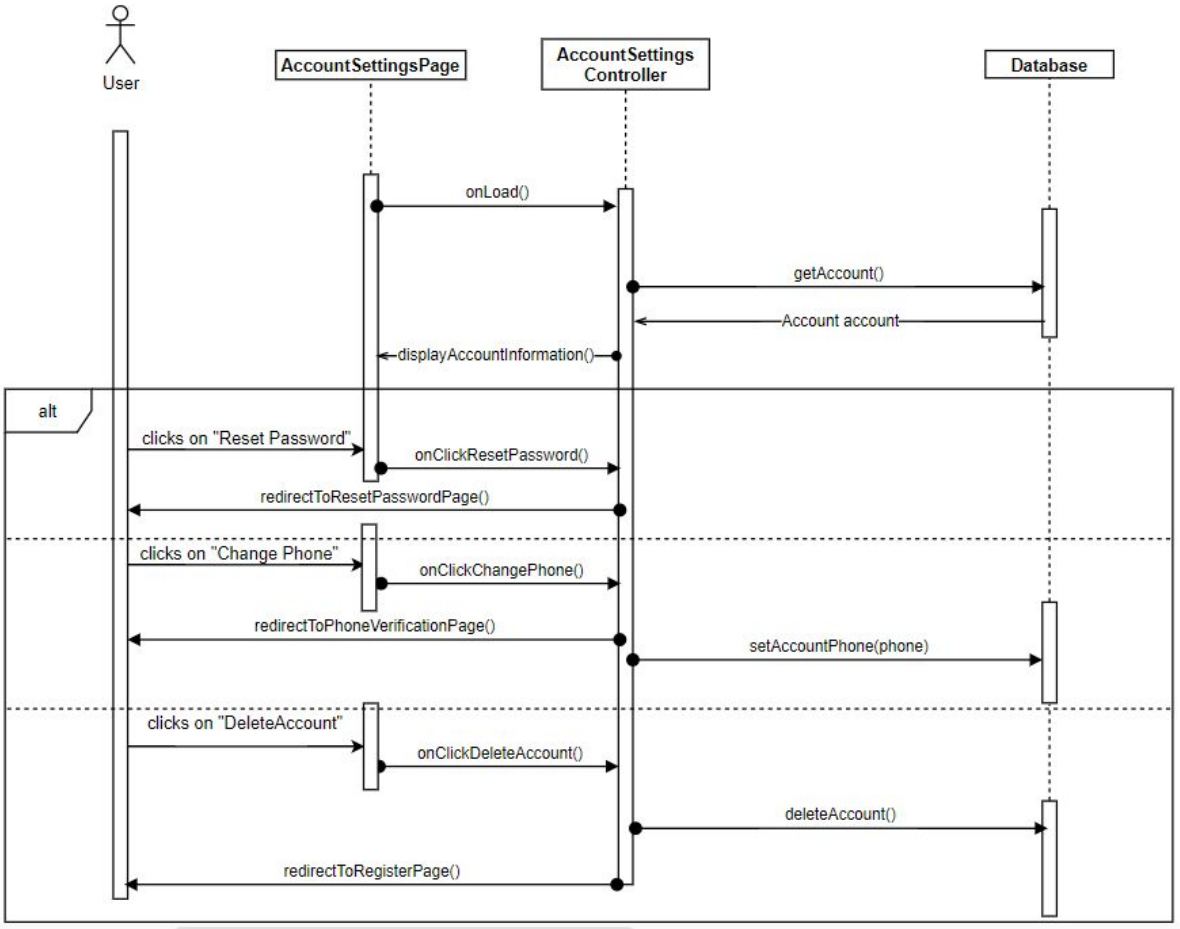
- User enters email password name information.
- User can start the authentication process by clicking on the “signUp” button which redirects to the phone verification page to continue with the process.
- User enters their phone number and verifies it and is redirected to the login page.
- If verification is unsuccessful, the user is redirected to the registration page.
- Gradschool and Department users’ accounts are registered to the system by the system admin manually.

4.3.2 Login Interaction Viewpoint



- User can enter their email and password, click the login button.
- User can click the google login button.
- User can click the facebook login button.
- User get authenticated by firebase auth. If unsuccessful, the user is notified.
- Account information fetched from database and user redirected to the dashboard.

4.3.3 Account Settings Interaction Viewpoint



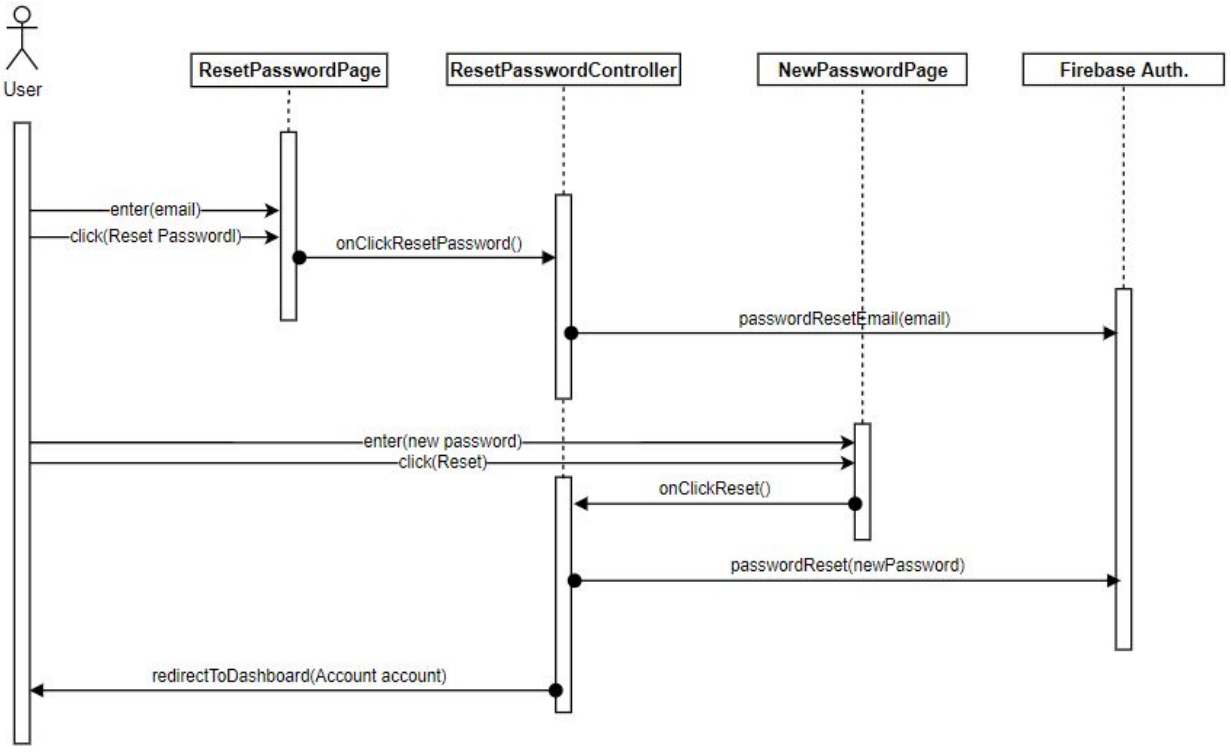
- User's account information is fetched from the database and shown to the user.

- User can start the reset password process by clicking on the "Reset Password" button which redirects to the reset password page to continue with the process.

- User can change their phone by clicking on the "Change Phone" button which redirects the user to the phone verification page to verify the new phone number, if so, sets the new phone on the database.

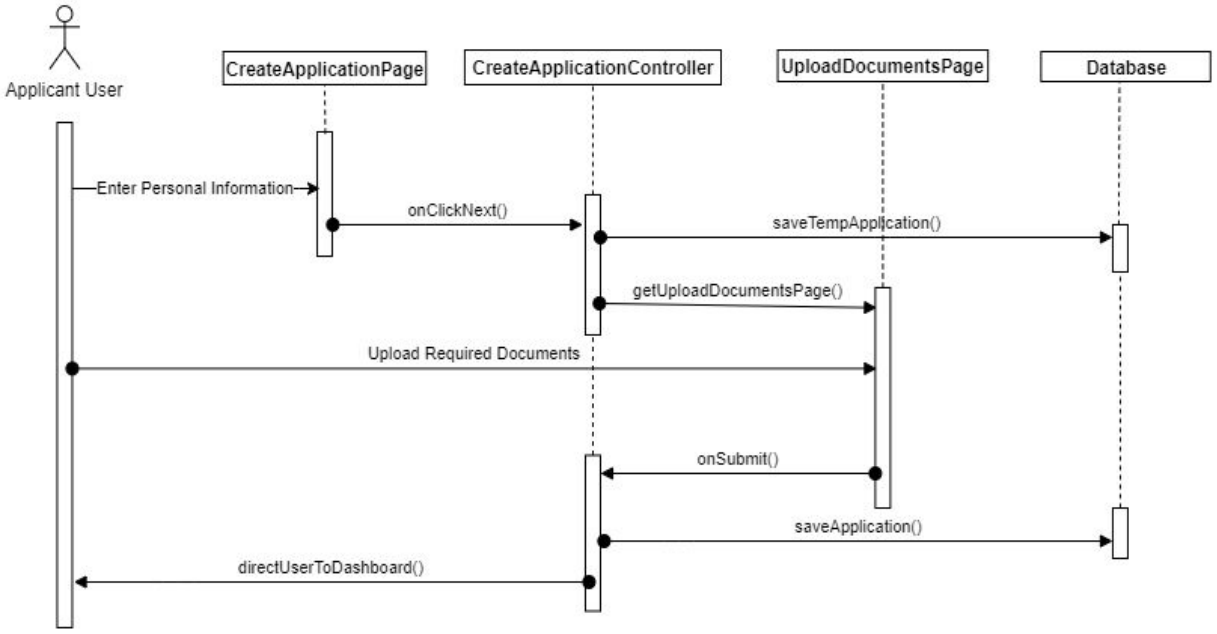
- User can start delete account process by clicking "Delete Account" button which deletes the account from database. User is redirected to register page after the deletion is done.

4.3.4 Reset Password Interaction Viewpoint



- User enters his/her account email and clicks on “Reset Password” button.
- Password reset email send via Firebase Authentication.
- User clicks the link at the email send, New Password Page comes up.
- User enters a new password, then clicks on the “Reset” button.
- New password set on Firebase Authentication.
- At the end user is redirected to his/her dashboard.

4.3.5 Create Application Interaction Viewpoint



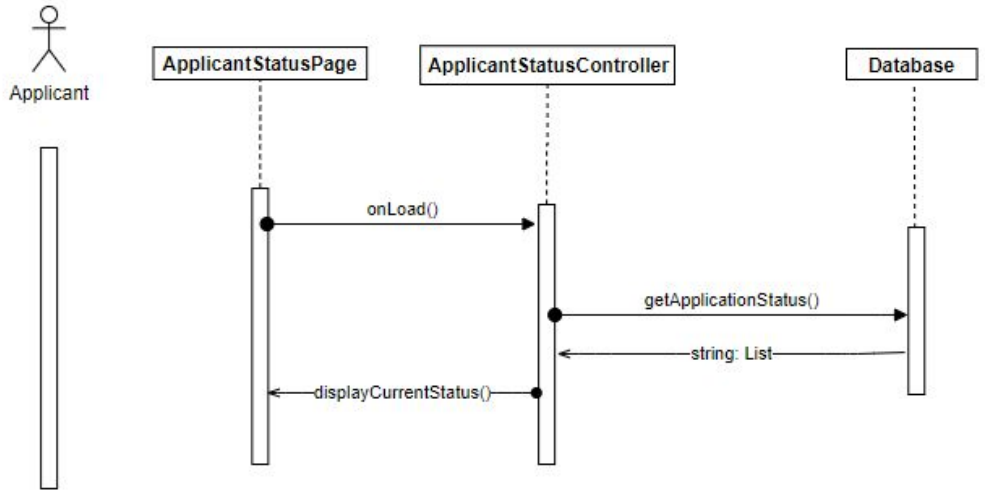
- After the user has entered his/her personal information for applying process and pressed the “next” button, CreateApplicationController gets the necessary information about the user from CreateApplicationPage through onClickNext() function and it saves the information of the user’s application to the database.

- When saving to the database is completed, CreateApplicationController redirects the user to UploadDocumentsPage by calling getUploadDocumentsPage() function.

- Whenever the user uploads required documents and presses the “submit” button, CreateApplicationController gets the necessary information about the user's documents from UploadDocumentsPage through onSubmit() function.

- After application is completed, CreateApplicationController saves all information, including documents, of the user’s application to the database and redirects the user to his/her dashboard.

4.3.6 Application Status Interaction Viewpoint

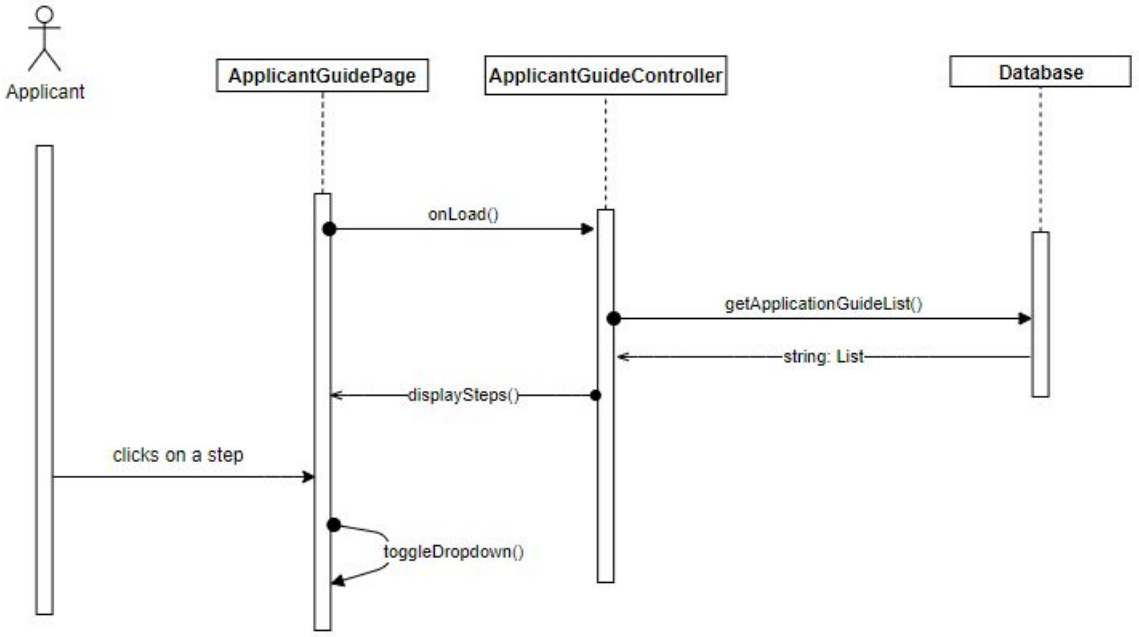


- ApplicantStatusPage sends an “onLoad" function request to the ApplicantStatusController.

- ApplicantStatusController gets the information about whether the application of the applicant has been confirmed or not, including interview information, from the database.

- ApplicantStatusController displays the application status on ApplicantStatusPage.

4.3.7 Applicant Guide Interaction Viewpoint

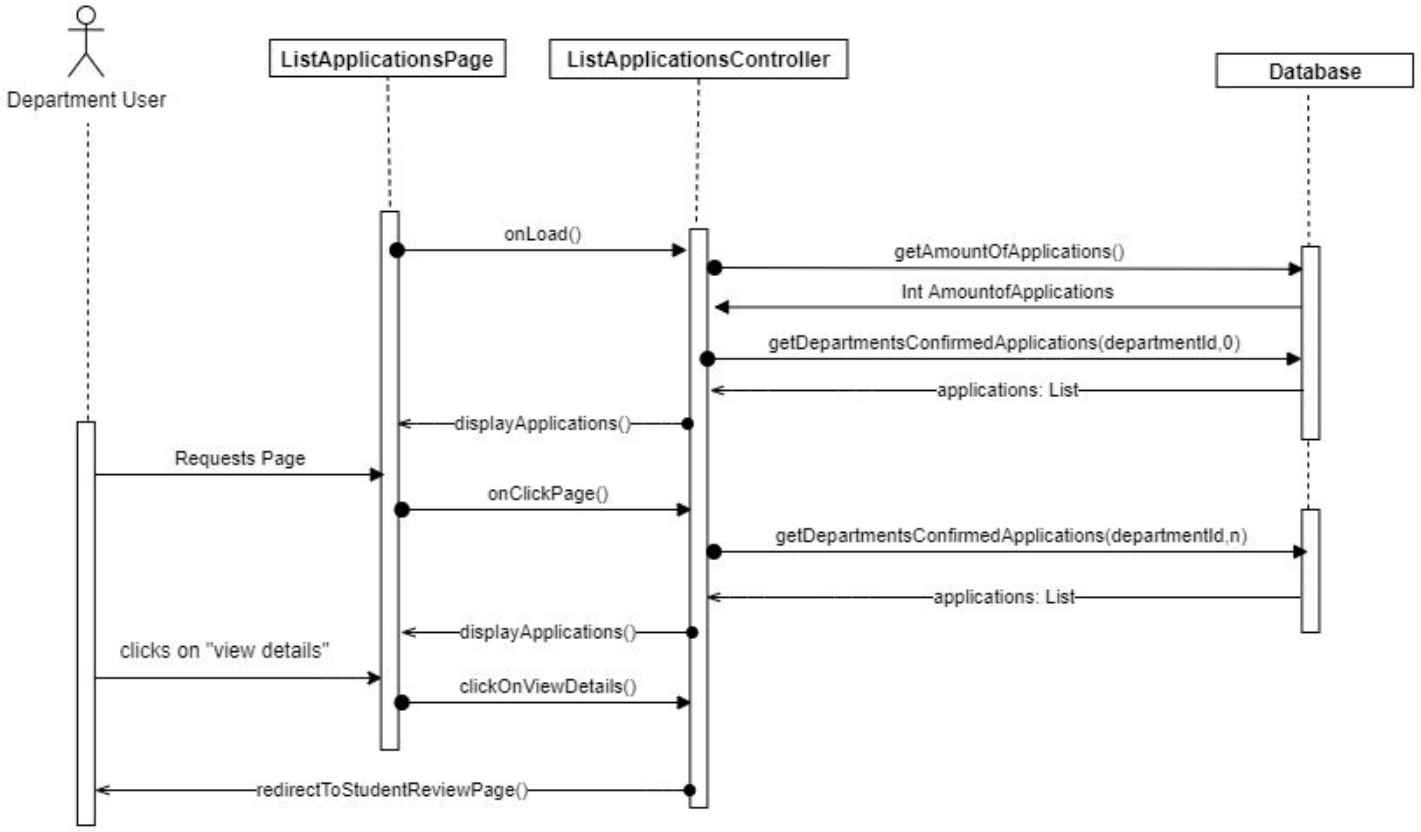


- ApplicantGuidePage sends an “onLoad" function request to the ApplicantGuideController.

- ApplicantGuideController gets the Application Guide as a list from the database and displays the steps of the guide on ApplicantGuidePage.

- Whenever the Applicant clicks on a step, the corresponding step will be shown to the applicant by requesting the toggleDropDown() function.

4.3.8 Department's List Applications Interaction Viewpoint

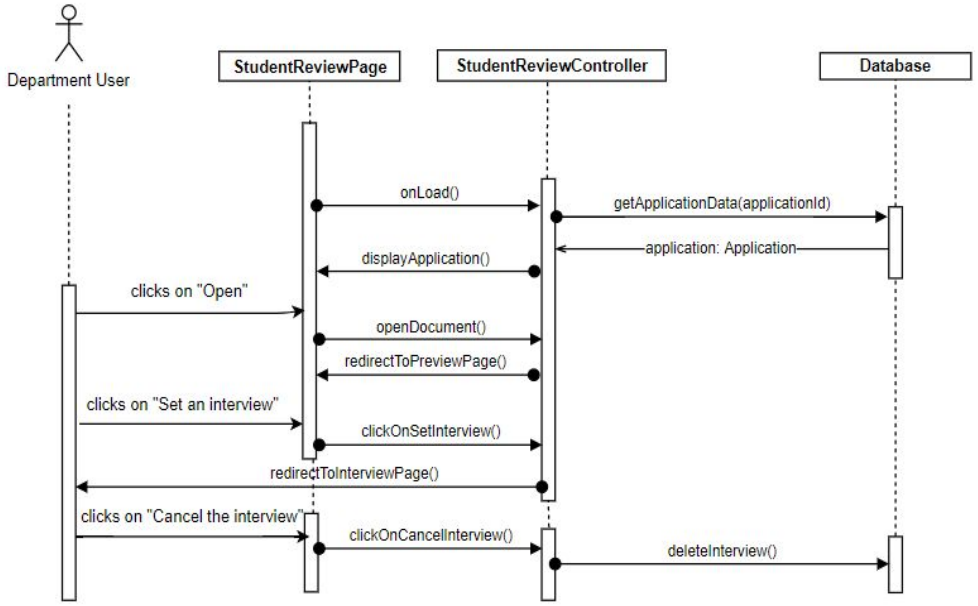


- ListApplicationsController initially gets the first 20 applications from the database for the first page (pageNo = 0) and displays it on the ListApplicationsPage.

- Whenever the user clicks on a page number or next/previous buttons ListApplicationsController gets the corresponding applications from the database as a list, and displays them on ListApplicationsPage.

- If the user clicks on 'View Details' of a particular application, then the user is redirected to the review page of that application.

4.3.9 Student Review Interaction Viewpoint

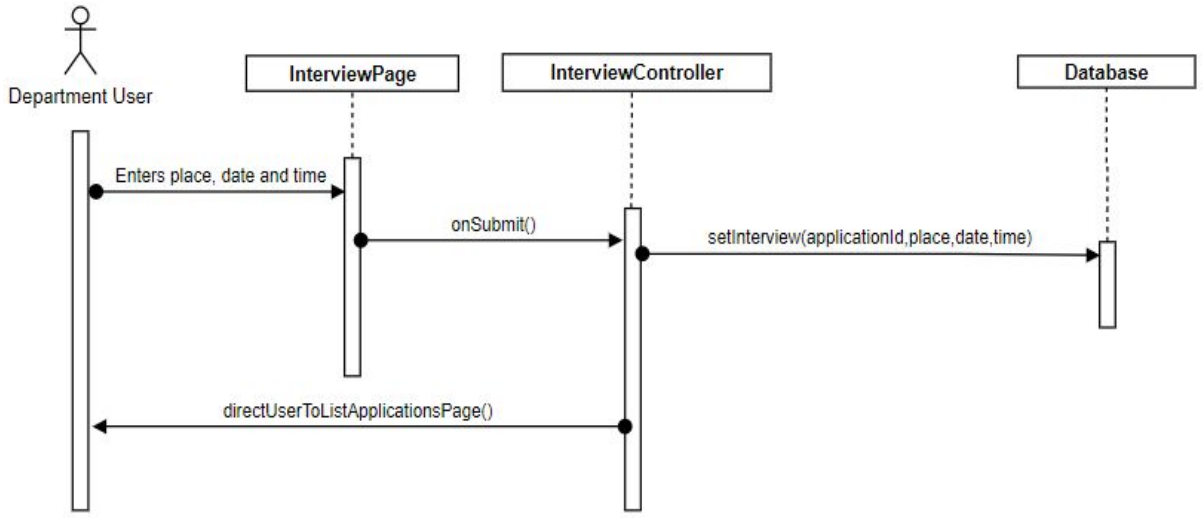


- StudentReviewPage requests the application data via StudentReviewController from the database, StudentReviewController gets the data and displays it on the StudentReviewPage.

- Whenever the user clicks on one of the listed documents' 'Open' button, StudentReviewController opens up that document on a new tab from the path it had fetched from the database when the page initially requested application data.

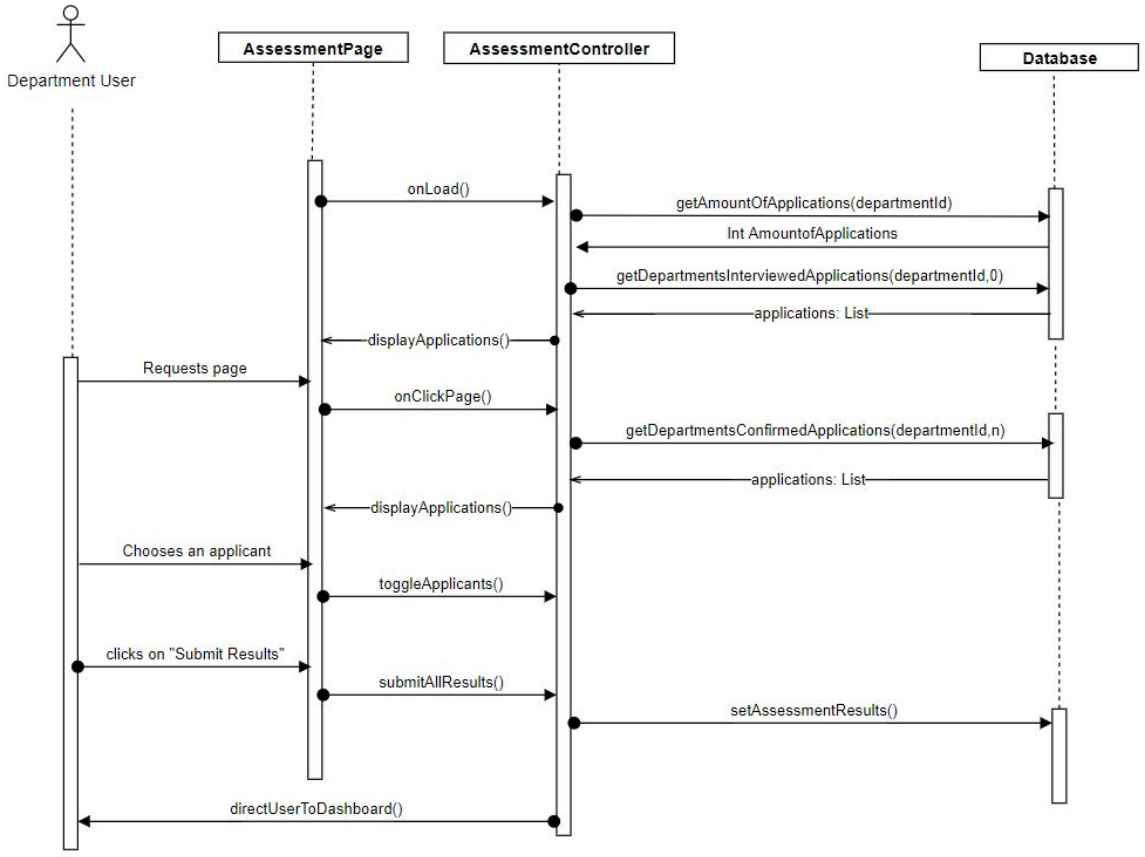
- If the user clicks on 'Set an interview' button, he/she will be directed to the Interview page, if, however, 'Cancel the interview' button is clicked, StudentReviewController resets that application's interview data fields.

4.3.10 Interview Interaction Viewpoint



- User enters the date, time and the place for the interview.
- Then the InterviewController changes the applicant’s interview data to whom the application belongs.
- When it's done, the user is redirected back to the list applications page.

4.3.11 Enter Assessment Results Interaction Viewpoint



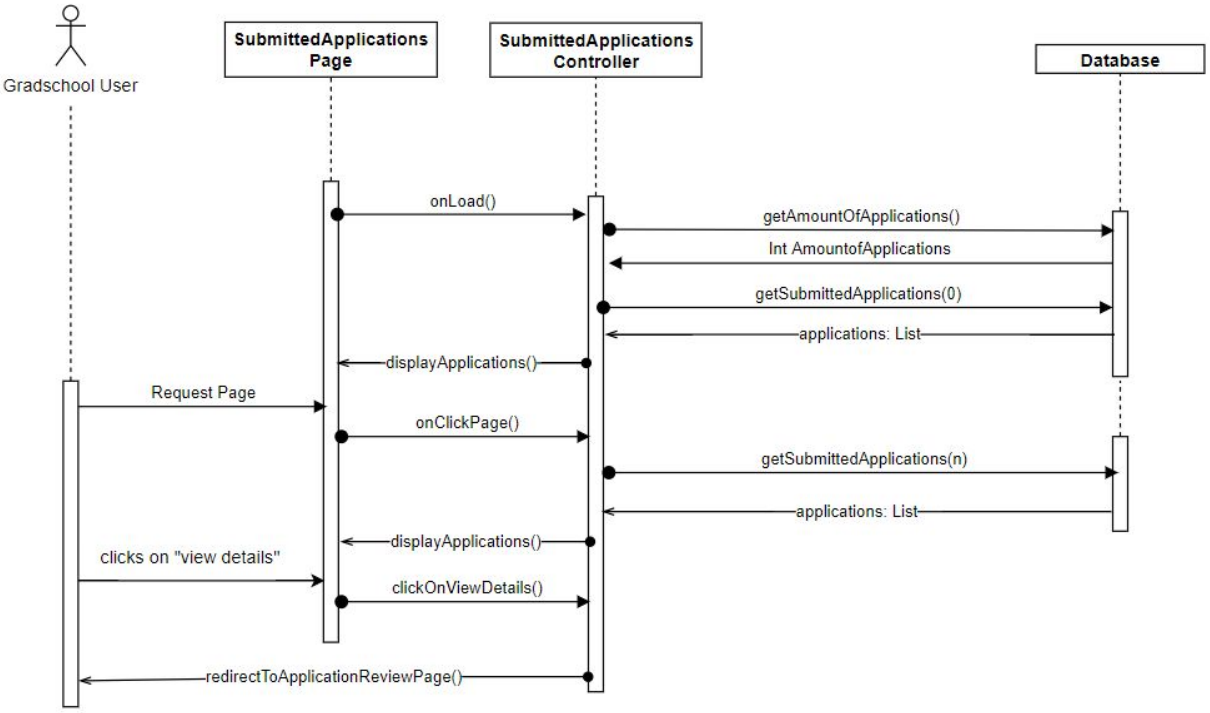
- AssessmentPage sends an “onLoad" function request to the Assessment Controller.

- AssessmentController gets the interviewed applications, whose page number is 0,according to Department Id from the database.

- Whenever the Department wants to view interviewed applications on any page number, the AssessmentController receives the interviewed applications for that page number from the database.

- After the Department submits the assessment results of the selected applications, the assessment results are sent to the database by AssessmentController.
- After the assessment results are sent, the Department is directed to his/her dashboard by AssessmentController.

4.3.12 Grad School's Show Submitted Applications Interaction Viewpoint

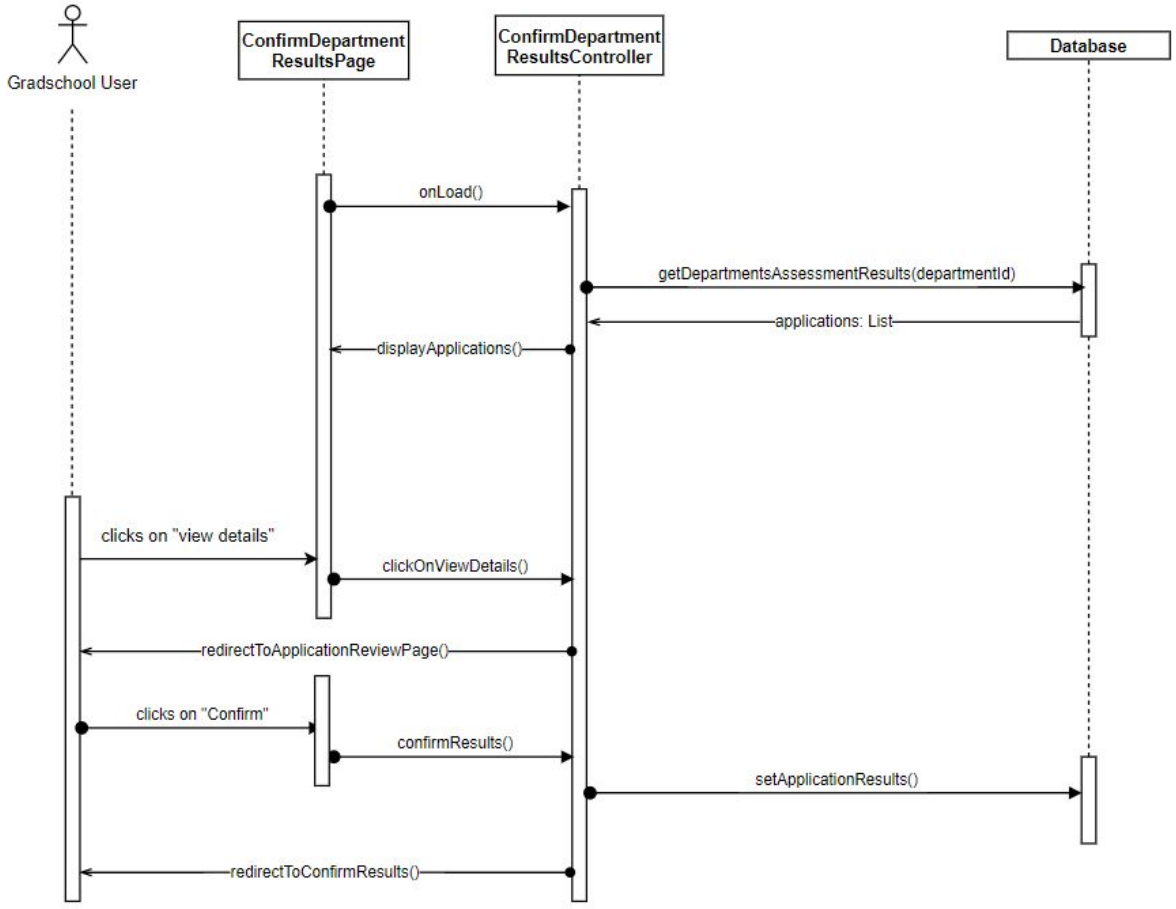


- SubmittedApplicationsController gets the first 20 applications from the database for the first page (pageNo = 0) and displays it on the ListApplicationsPage.

- Whenever the user clicks on a page number or next/previous buttons SubmittedApplicationsController gets the corresponding applications from the database as a list, and displays them on SubmittedApplicationsPage.

- Whenever the user clicks on the "view details" button SubmittedApplicationsController redirects the user to the ApplicationReviewPage.

4.3.13 Confirm Department's Results Interaction Viewpoint

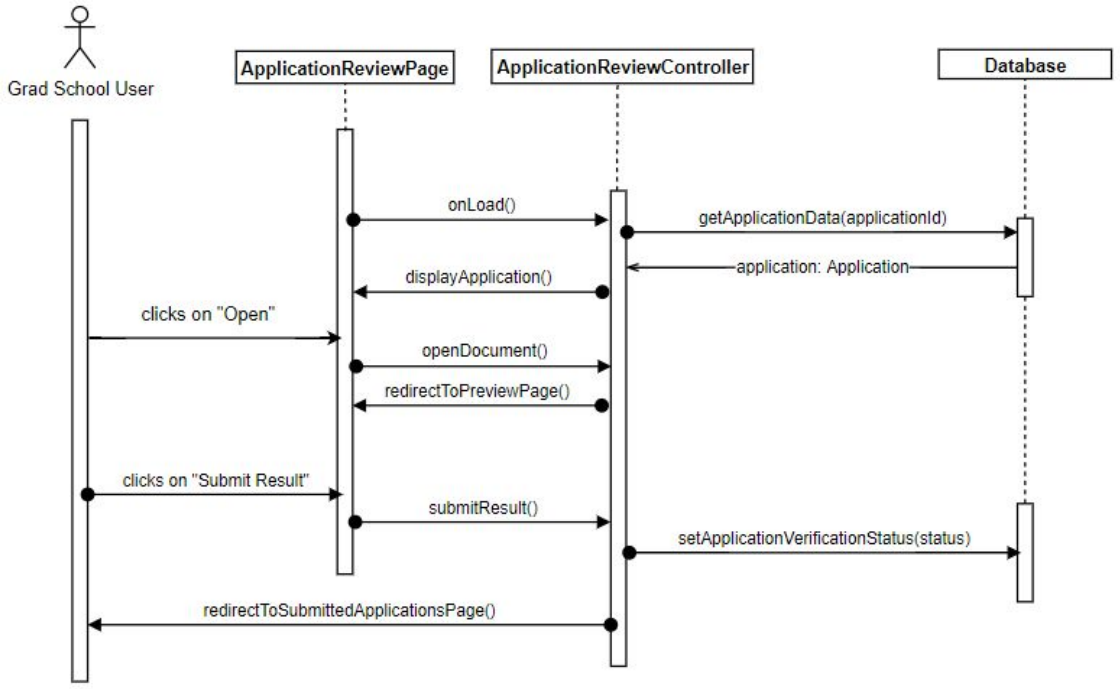


- ConfirmDepartmentResultsController gets the assessment results of the given departmentId as a list then displays it to the user.

- Whenever the user clicks on the "view details" button of a listed application result, ConfirmDepartmentResultsController redirects the user to the ApplicationReviewPage.

- Whenever the user clicks on the "confirm" button ConfirmDepartmentResultsController sets the chosen applications' "result" fields "true" in the database. And when the process is done the controller redirects the user to the ConfirmDepartmentResultsPage which contains updated information.

4.3.14 Application Review Interaction Viewpoint

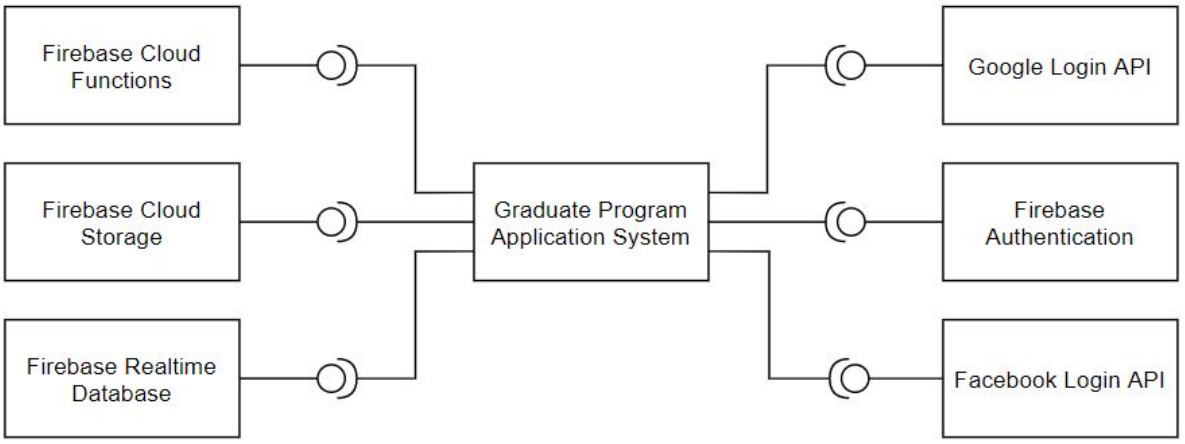


- ApplicationReviewController gets the corresponding application's data with given applicationId and displays it to the user at the ApplicationReviewPage.

- Whenever the user clicks on the "Open" button of the given document. ApplicationReviewController redirects the user to the PreviewPage which the user can see the content of the document.

- Whenever the user clicks on the "Submit Result" button ApplicationReviewController sets the current application's "isVerified" field with the given integer(status) and it redirects the user to the SubmittedApplicationsPage.

4.4 Interface Viewpoint



4.4.1 Firebase Cloud Functions

- Used for microservices (HTTP endpoint for serverless backend).
- Handles any type of scheduled tasks.
  Take backup of database and store in a secure cloud.
- Handles any type of trigger based tasks.
  - e.g. Delete the user account from the database when the user is removed from firebase authentication.
- Handles client triggered methods
  - e.g. Send email to applicants.
- Scales up the system's resources to match the user's usage patterns to provide a better performance.
- Requests will be executed with GET and POST HTTP requests on the URL of the function. Any additional data will be placed to the body of the request. This way, methods will be determined at the implementation phase by the developer as they are needed.
- Used version: 3.6.1

4.4.2 Firebase Cloud Storage

- Used for storing applicants' documents in the application.
- Used when creating an application or viewing an application.
- Sends application id to fetch documents.
- Used version: 7.14.4

Methods

for uploading application documents and profile pictures of users.

- put(file,metadata):
    - uploads file to storage.
    - file is type of File
    - metadata is a map
    - result handled by callback functions thus it returns void.

for getting the download url of application documents and profile pictures of users.

- getDownloadURL():
    - gets the access link to the file.
    - result handled by callback functions thus it returns void.

for giving a path to cloud storage specifying where the data should be.

- storageRef.child(path):
    - return reference to the given path.
    - path is string value of reference
    - result handled by callback functions thus it returns void.

4.4.3 Firebase Realtime Database

- Used in almost every process.
- Send user id to fetch user data.
- Can get application data.
- Used version: 7.14.4

Methods

for giving a path to the database specifying where the node should be.

- database().ref(path):
    - path is string value of reference
    - returns reference to the db position.
    - result handled by callback functions thus it returns void.

for creating an account and application data on the database.

- set(value):
    - sets a node to given value
    - value can be any primitive JSON type or some maplike custom type.
    - result handled by callback functions thus it returns void.

for deleting accounts from the database.

- remove():
    - deletes referenced data
    - result handled by callback functions thus it returns void.

for updating application data by grad school and department, user settings changes.

- update(Map):
    - simultaneously write to specific children of a node without overwriting other child nodes
    - result handled by callback functions thus it returns void.

for getting a node's value from the database. Will be used for almost every use case.

- once('value'):
    - get the value of reference once.
    - result handled by callback functions thus it returns void.

4.4.4 Firebase Authentication

- It is used in Login, Register and password reset processes.
- Sends user email and password returns boolean.
- Send user email, send user a password reset email.
- Used version: 7.14.4

Methods

- firebase.auth().createUserWithEmailAndPassword(email, password):
    - takes email and password. Both are String type.
    - error will be handled with catches if no error is thrown it will be passed as success.
- firebase.auth().signInWithCredential(credential)
    - result handled by callback functions thus returns void.
- firebase.auth().signOut()
    - result handled by callback functions thus returns void.

For Google login integration to Firebase authentication:

```
var credential = firebase.auth.GoogleAuthProvider.credential(
        googleUser.getAuthResponse().id_token);
```

For Facebook login integration to Firebase authentication:

```
var credential = firebase.auth.FacebookAuthProvider.credential(
        event.authResponse.accessToken);
```

4.4.5 Google Login API

- It is used in Login and Register processes.
- Sends no data to process, only redirects users to google services to get their credentials.
- Gets user credentials to authenticate users.
- Used version: 51.0.0

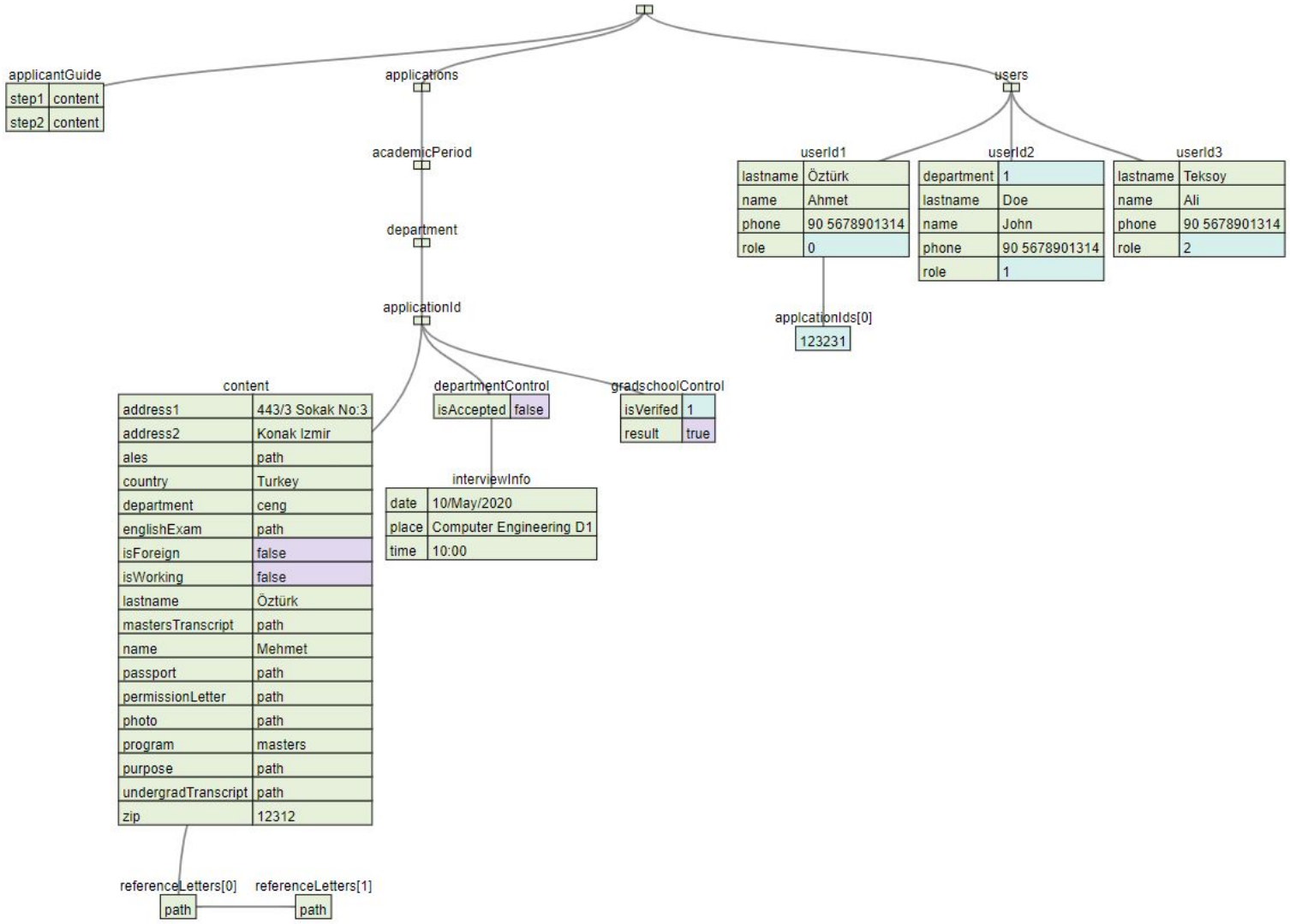
  Methods:
  - html class "g-signin2" : handles the login part itself, nothing is shown to the developer.
  - signOut(): terminates user session.

4.4.6 Facebook Login API

- It is used in Login and Register processes.
- Sends no data to process, only redirects users to facebook services to get their credentials.
- Gets user credentials to authenticate users.
- Used version: 7.0

  Methods:
  - FB.logout(function(response))
  - FB.login(function(response))

4.5 Data Model Diagram



This is a tree diagram of our NoSQL database. Entity Relationship Diagram is not applicable for NoSQL databases, so to model and visualize the database we draw a tree diagram which shows the key-value pairs of the database.

users: Holds unique userId key-value pairs for every registered user.

userId: Unique userId key, holds user information.

- lastname: (String) Last name of the user
- name: (String) Name of the user
- phone: (String) Phone number of the user
- role: (Int) Represents the user's role in system; applicant: 0, department: 1, gradschool: 2

- departmentId: (int) Represents department the user with department role belong. Only exists if the user is a department role user.
- applicationIds: (List<Integer>) Keeps the application id(s) of the user.

applications: Holds academicPeriod key-value pairs.

academicPeriod: Groups applications by academic period, holds department key-value pairs.

department: Groups applications by academic period, holds userId key-value pairs.

applicationId: Unique id for each application.

content: Holds an application's content

- address1: (String) Address line 1
- address2: (String) Address line 2
- zip: (String) Address zip code
- country: (String) Country info
- department: (String) target department of application
- isForeign: (Boolean) Applicant foreign or not foreign
- isWorking: (Boolean) Applicant is working or not working
- lastname: (String) Applicant last name
- name: (String) Applicant name
- program: (String) Intended program
- ales: (String) Document path
- englishExam: (String) Document path
- mastersTranscript: (String) Document path
- passport: (String) Document path
- permissionLetter: (String) Document path
- photo: (String) Document path
- purpose: (String) Document path
- "undergradTranscript": (String) Document path
- referenceLetters: List of document paths (Might be multiple)

departmentControl: Holds key-value pairs that only department users modify.

interviewInfo: Holds key-value pairs date, place, time.

- date: (String) Date information of the interview.
- place: (String) Place information of the interview.
- time: (String) Time information of the interview.
- IsAccepted: (Boolean) Applicant entered the assessment list or not.

gradSchoolControl: Holds key-value pairs that only grad school users modify.

- IsVerified: (Boolean) Application approved by grad school or not.
- result: (Boolean) Final result of the application.

ApplicantGuide: Holds key-value pairs for each step to represent in applicant guide.

4.6 Design Rationale

Unified Modeling Language (UML) of version 2.5.1 is used in making design viewpoints.

In the logical viewpoints, class diagrams are made to design the static structure of the program by using the MVC design pattern which will yield a much faster and better implementation phase.

MVC is convenient in designing web applications since it enables the application to provide multiple views, support Javascript's asynchronous methods while providing low modification cost whenever a change is requested.

In the interaction viewpoints, sequence diagrams are used for explaining the flow of the system and giving a full understanding about what users are capable of doing by using this application.

In the interaction viewpoints, some of the methods instead of returning a boolean value that would indicate the success or failure of the operation, will execute differently predefined callback functions depending on the success/failure of the method.

The software is designed using MVC to make the product extensible, and maintainable by dividing each case into three basic components; a view that represents the interface which interacts with the user (inputs, buttons, form fields, etc.), a controller that controls the view by handling the inputs/requests from the user through that view, interacts with the database and updates the view whenever necessary, finally a model that is a datawise representation of how the controller will handle requests.

NoSQL(non-sql, non-tabular, non-relational) database is used hence more of a key-value schematic data model is used instead of an entity-relation diagram. Because ERD is not capable of fully expressing how our database will be structured. That's why we are using a data model to show how our database is structured.

In the implementation phase, cost of different operations using NoSQL is calculated to be less than a relational database (SQL) that is because the structured approach of relational database like SQL slows down performance as data volume or size gets bigger and it is also not scalable to meet the needs of ‘Big Data’.

In the interface viewpoint, interface diagrams are made to show which external services our application will use, below those diagrams are their descriptions of why a particular service is needed and which functions will be used to interact with them.

Interface diagrams are made using the lollipop notation since each of these service providers have an interface that enables our program to use necessary functions.

System exploits event-driven architecture to make it loosely coupled since event creators will not be aware of which event consumers are listening for an event and the event doesn’t know what the consequences are of its occurrence.

Methods and attributes in the class and sequence diagrams are very indicative of their functionalities as their names clearly express what they do, making the diagrams and the design of the software easy to read.