DukeStudy Documentation

Team JBeibS

John Bralich, Branch Vincent, Sravya Chelikani

Application Description

DukeStudy is a mobile Android application designed to help connect Duke University classmates in order to produce a more productive learning environment. Users have the primary ability to find and add classes offered at Duke for the current academic semester. Within each class, users can view or post questions, view or create study groups, and view a list of classmates. Within a study group, users can also view or post questions as well as view or schedule a study event.

1 Contents

2 Revision History

Date	Revision
4/16/2017	Creation
4/17/2017	Added Overview, Architecture, Project Plan, Test Methodology, Test Plan, and Schedule
4/19/2017	Added Revised Requirements Section
4/19/2017	Final Version

3 Overview

Below, we give a brief overview of the application, including the purpose and scope.

3.1 Description

DukeStudy is a mobile Android application designed to help connect Duke University classmates in order to produce a more productive learning environment. Users have the primary ability to view and add classes offered at Duke for the current academic semester. Within each class, users can view or post questions, view or create study groups, and view a list of classmates. Within a study group, users can also view or post questions as well as view or schedule a study event.

3.2 Purpose

The purpose of this application is to help connect Duke University classmates, to produce a more productive and collaborative learning environment.

3.3 Scope

Our intended consumer is Duke students looking for study groups. As such, the courses included in our course listings would be an exhaustive list of current duke courses.

3.4 Glossary

1. **User** – the student who will actually interact with the application.
2. **Use case** – describes a goal-oriented interaction between the system and an actor. A use case may define several variants called scenarios that result in different paths through the use case and usually different outcomes.
3. **Actor** – user or other software system that receives value from a user case.
4. **Firebase** – mobile and web application platform with tools and infrastructure designed to help developers build high-quality apps. This included tools for database and authentication
5. **API** – Application program interface; specifies how software components should interact.
6. **Gantt Chart** – a chart in which a series of horizontal lines shows the amount of work done or production completed in certain periods of time in relation to the amount planned for those periods.

7. **Sprint –** In software development, a sprint is a set period of time during which specific work has to be completed and made ready for review.

4 Requirements

This section features the requirements for the application. First, we discuss the functional requirements, each of which includes the associated user requirement. Then, we overview the non-functional requirements.

4.1 Functional Requirements

UR1. User Identity
Biographical information *shall* be stored and displayed for each user.

> *SR1.1* The application *shall* allow previously unregistered users to create a new account by using the sign-up button. The user *shall* enter a Duke University email address and a password, at least 6 characters long, to register an account.
>
> *SR1.2* The application *shall* authenticate previously registered users to login to their accounts by entering the email id and the user's previously registered password and by clicking a login button.
>
> *SR1.3* Users that forgot previously registered passwords *shall* be able to reset their passwords by clicking on the Reset Password button and entering their registered email ID. Upon clicking this button, the application *shall* send an email to the registered email through firebase authentication, and allow users to set new password.
>
> *SR1.4* The user *shall* remain logged into the application till the user chooses to logout using the option in the right hand size menu of the application or presses the back button on the application.

UR2. User Interaction
The user *shall* interact with the application using a side navigation bar that contains all the options available to the user.

> *SR2.1* Side Navigation Bar: The side navigation bar *shall* contain the email and name of the user on it's header. It *shall* contain the option to add a class. The side navigation bar *shall* display all the classes and groups added by a user.

UR3. User Profile
The user *shall* view a profile page containing information about the user. The user *shall* be able to edit their details and save it to their account.

> *SR3.1* Upon signing into the application, the user *shall* see the profile page by default.

SR3.2 The profile page *shall* contain details of the user such as, Name, Email, Major, Graduation Year and a profile picture.

SR3.3 The user *shall* be able to edit all of the above details using an Edit Profile button.

UR4. Add Class

The user *shall* view a list of classes and be able to add a class by selecting the Add Class option in the side navigation bar.

SR4.1 An add class option *shall* allow the user to see a list of classes offered by Duke ECE department.

SR4.2 The application shall add a class to the user's database using a "+" button and remove it using a "-" button.

SR4.3 The application *shall* check the users' added classes and display a plus or minus accordingly.

UR5. Classes

The user *shall* be able to view a list of added classes in the side navigation bar. The user *shall* be able to view a particular class page by clicking on it in the side navigation bar.

SR5.1 A class added by a user *shall* be added to the user's list of classes in the database.

SR5.2 Upon clicking on the class, the application *shall* retrieve a three tab layout containing the class Posts, Groups and Members of the particular class.

UR5.1 Class Posts

The user *shall* be able to post and comment on a post in an added class.

SR5.1.1 The application *shall* retrieve the list of posts of a particular class.

SR5.1.2 The application *shall* allow users to add a new post or comment on existing posts.

SR5.1.3 The posts and comments *shall* contain the text, name and profile picture of the user that posted it.

UR5.2 Group List: The user *shall* view a list of groups in the class and either add or remove them using a "+" and "-" button next to the group name respectively.

SR5.2.1 The application *shall* keep track of added groups and display them in the side navigation bar.

SR5.2.2 The application *shall* allow users to add a group to their list and display them in the side navigation bar or remove a group from their list.

UR5.3 Class Members

The user *shall* be able to view a list with email and name of other users who have added the same class.

SR5.3.1 The application *shall* retrieve information of other users who added the same class.

SR5.3.2 Upon clicking on any user's details, the application *shall* retrieve profile information of the user.

UR6. Groups

A user *shall* be able to add a group in a course and view it in the side navigation bar.

SR6.1 A group added by a user *shall* appear in the side navigation bar.

SR6.2 Upon clicking on the group, the application *shall* retrieve a three tab layout containing the class Posts, Events and Members.

UR6.1 Group Posts

The user *shall* be able to post and comment on a post in an added group.

SR6.1.1 The application *shall* retrieve the list of posts of a particular group.

SR6.1.2 The application *shall* allow users to add a new post or comment on existing posts.

SR6.1.3 The posts and comments *shall* contain the text, name and profile picture of the user that posted it.

UR6.2 Group Events

The user *shall* view a list of study events in the group and either add or remove them.

SR6.2.1 The application *shall* allow users to add a new event by requesting information about the name, date, time and location of the event.

SR6.2.2 The application *shall* allow users to add an event or delete an event from their account using a "+" and "-" button next to the event name respectively

UR6.3 Group Members

The user *shall* be able to view a list with email and name of other users who have added the same group.

SR6.3.1 The application *shall* retrieve information of other users who added the same group.

SR6.3.2 Upon clicking on any user's details, the application *shall* retrieve profile information of the user.

4.2 Non-Functional Requirements

The non-functional requirements are presented below.

NR1. Platform
The application *shall* support the Android mobile platform. Previous versions of Android *shall* be supported such that their combined user-base is greater than or equal to 95% of the user-base of all current Android versions starting with the most current release.

NR2. Performance and Memory
The application *shall* follow the performance and memory specifications as specified by the Android Application Store.

NR3. Cost
The application *shall* be released as a free application.

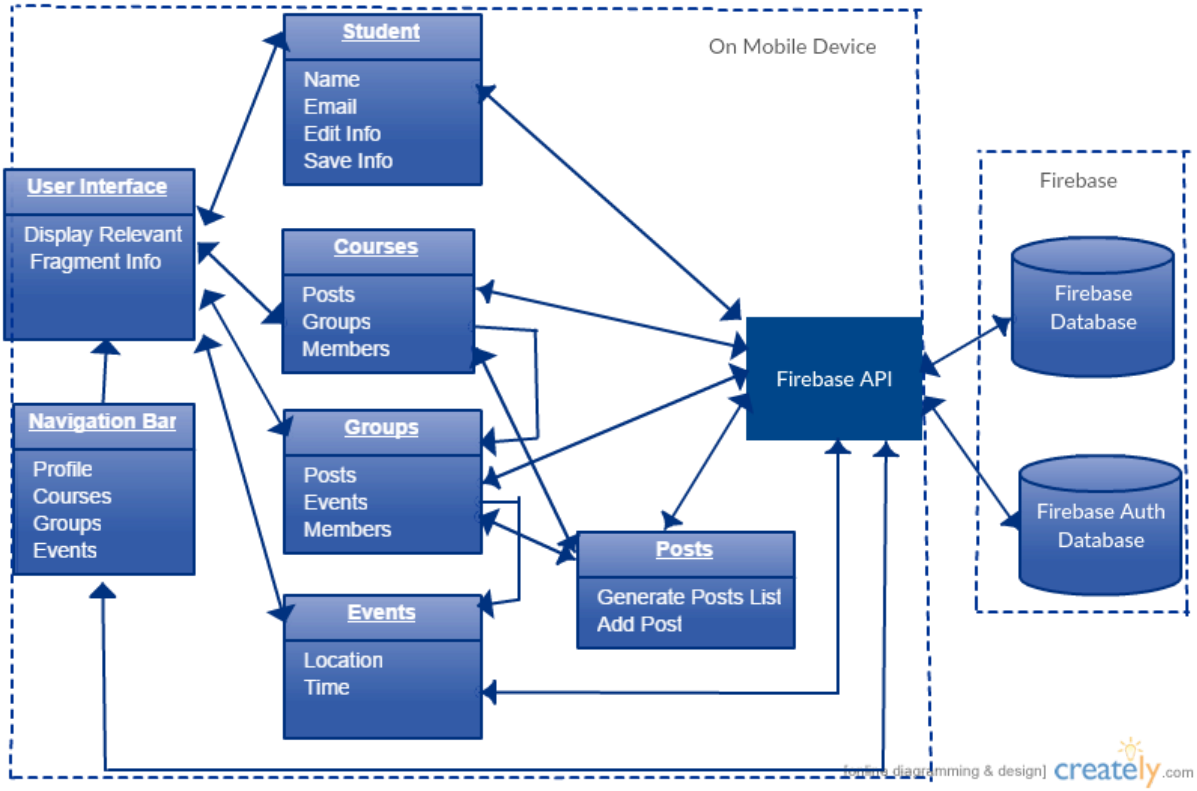
NR4. Release
The application *shall* be released by 4/21/2017.

NR5. Security
The application *shall* only be accessible to users that register with a Duke University email.

5 Architecture Model

5.1 System Architecture

The system architecture diagram is presented below. Note that the components in the dashed box reside on the mobile device and the components outside reside on an external device. The main components include the Navigation Bar, User Interface, Profile, Classes, Groups, Events, Posts, Firebase API, and Firebase Database along with Firebase Authentication Database. Single headed arrows indicate one way interactions while double headed arrows indicate two-way interactions between components.



5.2 User Story

Justin is a History major at Duke University who has to take a Computer Science class this semester. He knows very little about computers, much less science, and so he is very worried that he will not be able to pass. To make matters worse, he doesn't know anybody in the class, but prefers to study with a group, especially when he is out of his comfort zone. Luckily, Justin has DukeStudy. The application helps him connect with other students from his class and coordinate meeting times and locations for study sessions. From being able to post questions to join a group, Justin feels a little better about this semester.

When Justin opens the mobile DukeStudy application, he can expand the collapsible side navigation where he is presented with a menu of options: 'Profile', 'Classes', and 'Groups.'

If Justin selects 'Profile,' the application displays his public profile. The required information includes his first and last name, email, academic major, and graduation year. Optionally, he can add a profile picture.

Underneath 'Classes', the application lists the courses that Justin has previously added to his profile. At the bottom of this list, he can select the 'Add Class' option, which will present him with a list of classes offered during the current academic semester at Duke University. This list includes the course department, code and title. After he has found his Computer Science class, COMPSCI 101, he selects 'Add Class', which adds the class to his profile and thus to the side navigation.

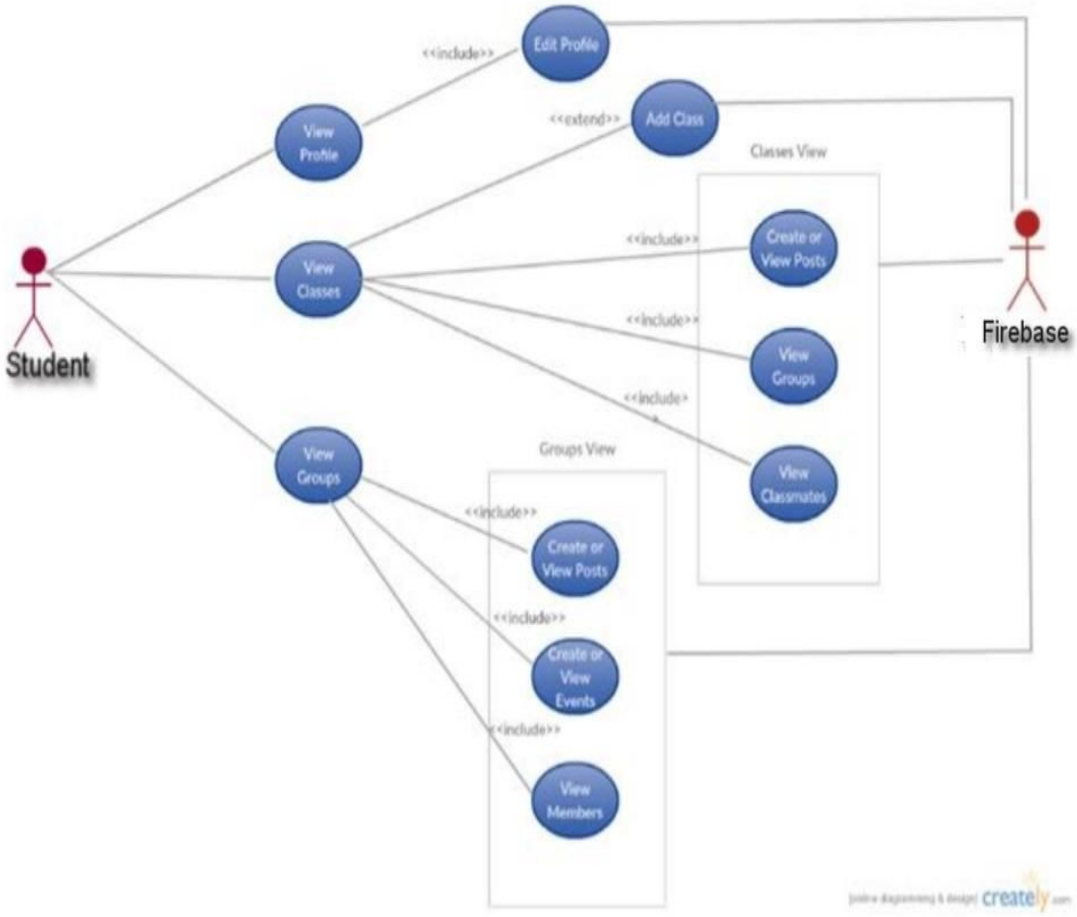
Now if Justin selects 'COMPSCI 101' in the side navigation, the application displays the course page. At the top of screen is a top navigation with three tabs: 'Posts', 'Groups' and 'Members.' The 'Posts' tab gives Justin the option to publicly post a question for all his classmates to see. Once posted, his classmates can respond with a 'Comment.' All posts are displayed in reverse chronological order. The 'Groups' tab lists the study groups created for that class. This list includes the group name and number of members. To the left of the group name, a "+" or "-" button allows him to either join a group or leave a group, respectively. At the bottom of the list, Justin can create a new study group. Finally, the 'Members' tab lists the full name of students that have added the class to their profile. If he selects a name, he will be taken to the student's public profile.

Once Justin has joined a group, he can select the group name under 'Groups' in the side navigation, taking him to the group page. At the top of screen is a top navigation with three tabs: 'Posts', 'Events' and 'Members.' The 'Post' tab gives Justin the same functionality as the course's 'Posts' tab, except these posts are now exclusive to the group. The 'Events' tab lists all scheduled study events. The list includes the event date, time, and meeting location. At the bottom of the list, Justin can create a new study event. The 'Members' tab is the same as the course's 'Classmates' tab, except only group members are listed.

5.3 Use Cases

5.3.1 Diagram

The below use case diagrams depict the actors, which are the user and the backend database. The use cases that describe the series of actions that can be taken by the actors are described below in tabular form. The associations between the actor and use cases are depicted by a line joining them.



5.3.2 View Profile

View profile allows a user to view his/her own individual profile.

View Profile	
Actors	User, Database
Description	User can view his profile details such as username and email ID, or edit his profile details and add them to the database, so that his profile can be viewed by other users of the application.
Data	Username, email
Stimulus	User clicks "Profile" button in the side navigation bar
Response	The right side of the screen shows the username being displayed and will present an "Edit Profile" option.
Comments	

5.3.3 View Classes

View classes allows the user to see the current list of classes that the he/she is enrolled in.

View Classes	
Actors	User, Database
Description	User can view his list of added classes and select any of them to view a Class page containing "Posts", "Groups" and "Classmates". The "Posts" button allows the user to post a question to members of the class and reply to posts. The "Groups" button allows the user to view all study groups in the class and join if necessary. The "Classmates" option allows the user to view profiles of students who have added the same class. If the user wants to add a new class "Add Class" option is provided, which retrieves the class list from database.
Data	User selects a class of interest and retrieves information of that particular class from database
Stimulus	User selects "Classes" button on the side navigation bar.
Response	The class page with options like "Posts", "Groups", "Classmates" pops up.
Comments	

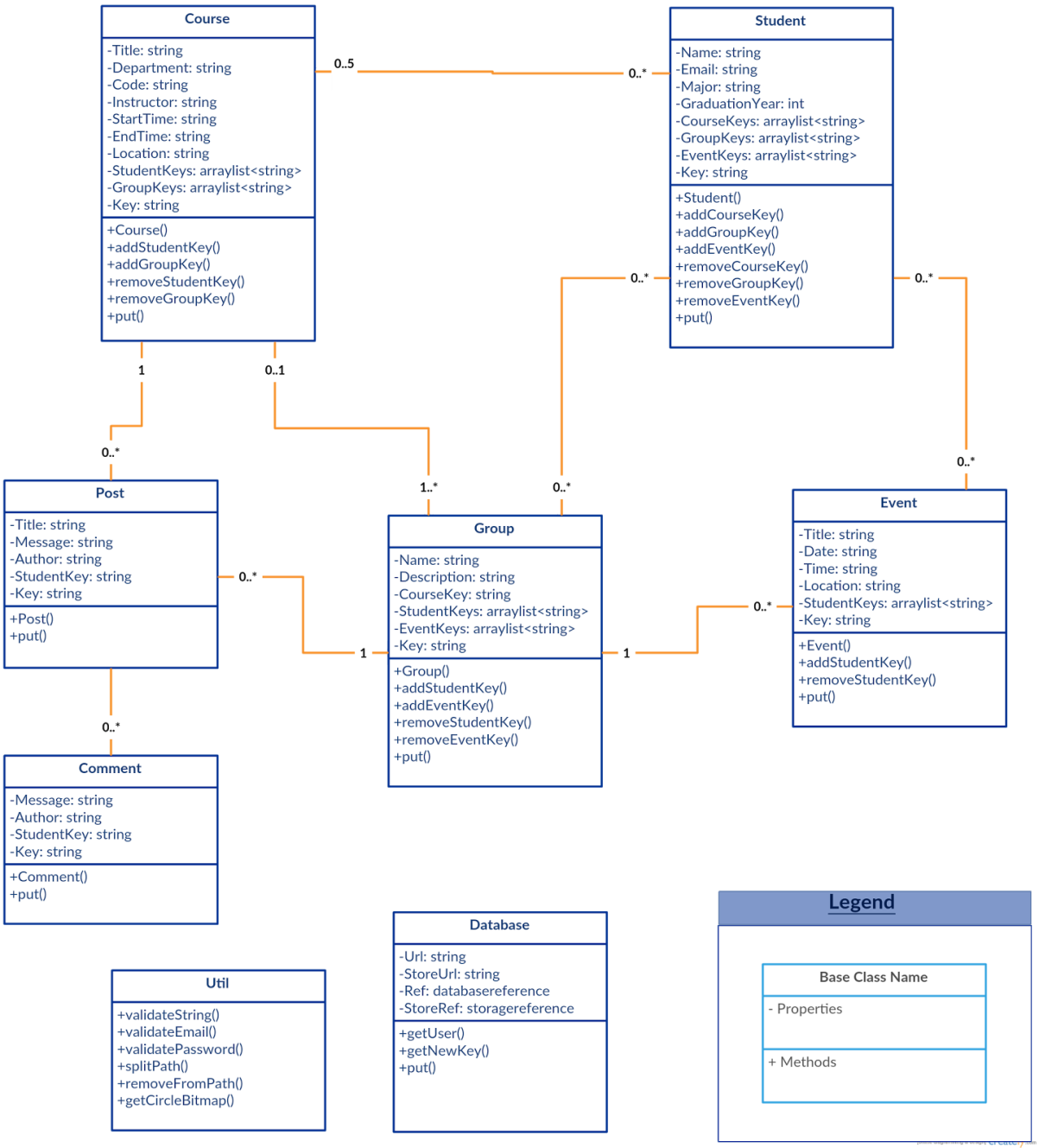
5.3.4 View Groups

View groups allows the user to see the current list of groups that he/she has joined.

View Groups	
Actors	User, Database
Description	The user can view a list of all the groups that he's a part of and select a group of interest to open up a Group page with options like "Posts", to post or reply to a question, "Events", to view upcoming study events of the study group and "Members" to view profile details of other users who are also in the current group.
Data	User sends a view current groups request and retrieves his current groups and the group page.
Stimulus	User selects the "Groups" option on the side navigation bar
Response	A list containing all the groups that the user is a part of, is returned. Selecting a group among these, returns the group page containing "Posts","Events" and "Members"
Comments	

5.4 Class Diagram

The Java class structure is outlined below. The main classes include Course, Student, Group, Event, Post, and Comment. Note that the inclusion of getters and setters for each class is not explicit, but rather implied. Also, our original class diagram used inheritance, but this was removed due to Firebase's restrictions.

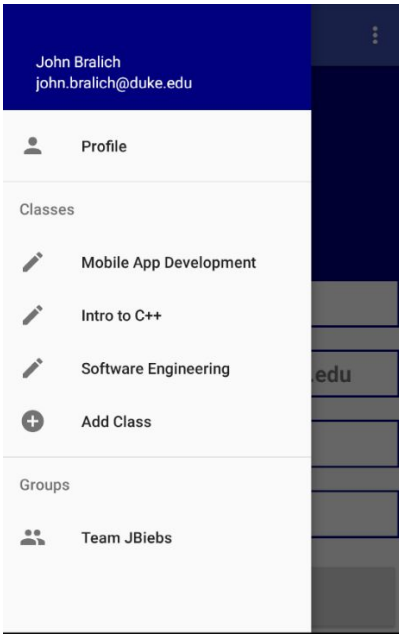


6 User Interface Design

In this section, we cover the basic User Interface design of our application for all of our various pages.

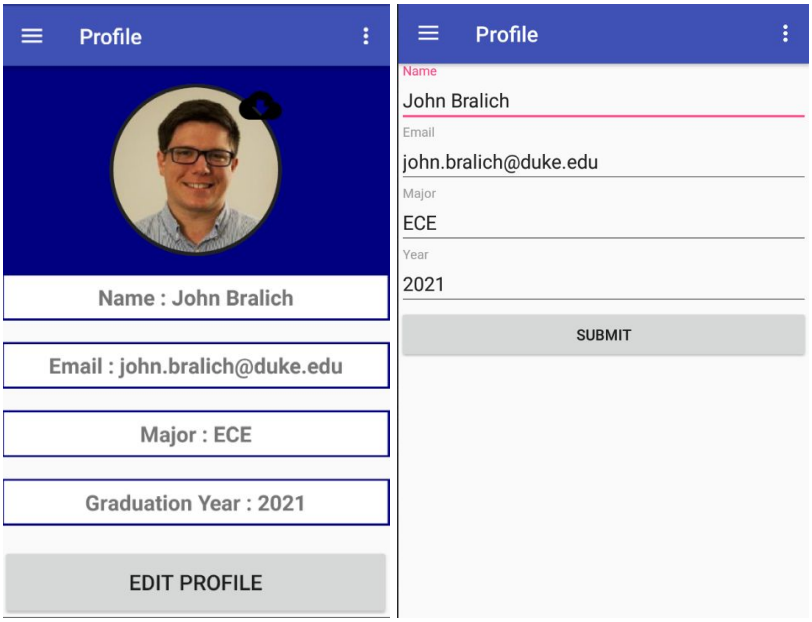
6.1 Navigation

The side navigation includes links to profile, course pages, and group pages.



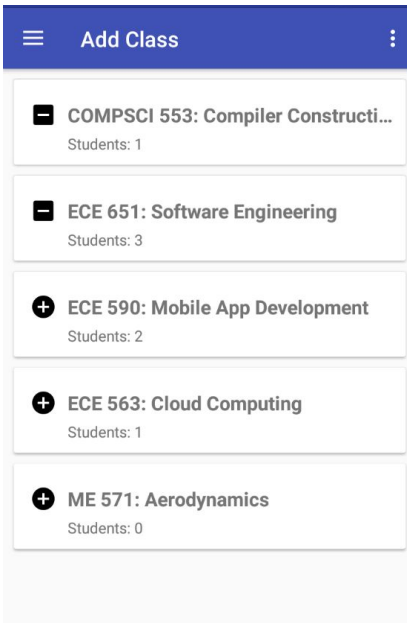
6.2 Profile

The profile page allows the user to view and edit his/her information.



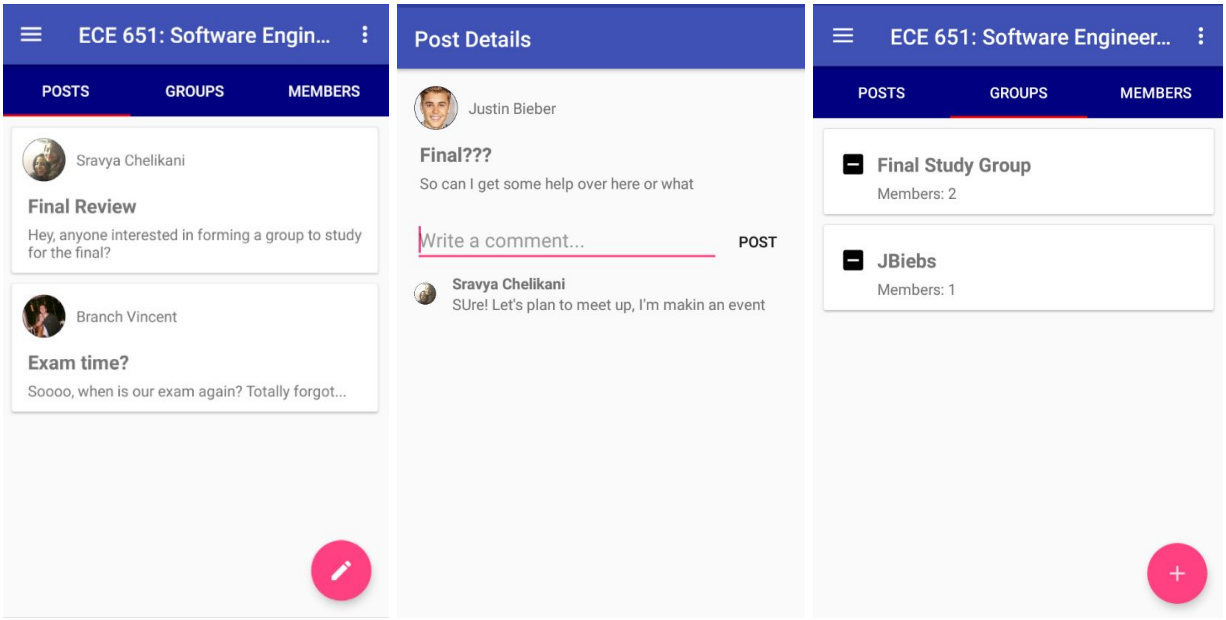
6.3 Add Course

The add course page allows the user to view and join classes.



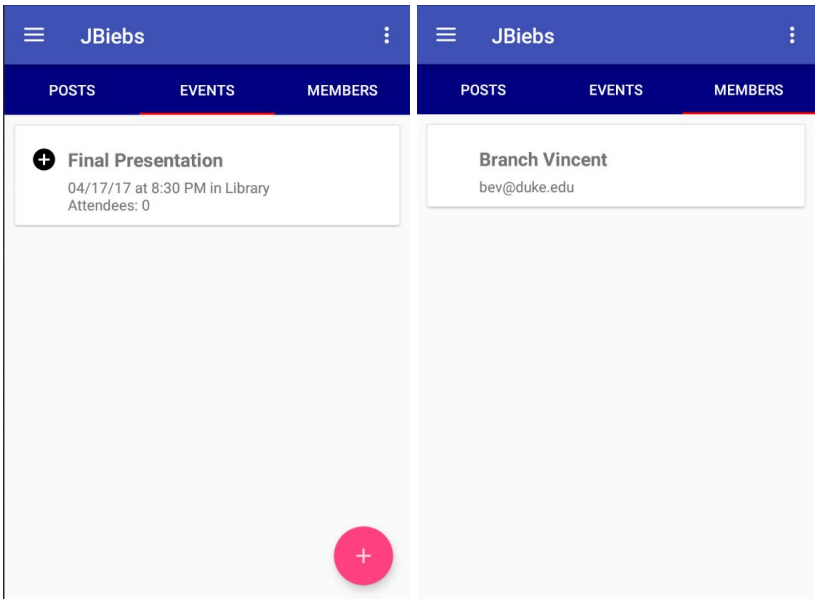
6.4 Course

The course page allows the user to post, join groups, and view classmates.



6.5 Group

The group page allows the user to post, joint events, and view group members.



7 Project Plan

In this section, we present our development and documentation plan, beginning on February 21 and ending on April 19. Changes to the original document are marked in red (deletions) and green (additions). The development plan is broken up into 3 separate sprints, each lasting 2-3 weeks. For each sprint, we present a detailed list of tasks including the dates and dependencies. The documentation plan follows a similar structure.

7.1 Sprint 1

In this sprint, we developed the basic User Interface design for the various pages as well set up our backend database for reading and writing simple dummy data. This sprint ran from February 21 to March 9. A detailed description of this sprint and associated tasks can be found below.

Task ID	Title	Description	Start	End	Duration (days)	Asignee	Dependencies
1.1	Side Navigation Bar	Create side navigation drawer that lists all page options.	2/21	2/22	2	John	
1.2	Top Navigation Bar	Create template for pages with top navigation bar. At least 3 navigation tabs will be included.	2/23	2/24	2	John	
1.3	Profile Page	Create template for profile page containing static, dummy data.	2/25	2/26	2	Branch	
1.4	Course Page	Create template for course page. Note this uses the top navigation template and will be filled with static, dummy data.	2/27	2/28	2	Branch	1.2
1.5	Add Course Page	Create template for add course page containing static, dummy data.	3/1	3/2	2	Branch	1.1
1.6	Groups Page	Create template for groups page. Note this also uses the top navigation template and will be filled with static, dummy data.	3/2	3/3	2	Sravya	1.2
1.7	Navigation Integration	Link options listed in navigation bar to page templates listed above.	3/4	3/5	2	John	1.1 1.3-6
1.8	Database Setup	Setup backend database using Firebase console and test reading/writing dummy data.	3/5	3/9	5	Sravya	

7.2 Sprint 2

In this sprint, we developed our application's backend by realizing the exact database structure and creating the core Java objects, as outlined in the class diagram. Additionally, we integrated the database with the Java objects and verified by performing tests with real class data. This sprint ran from March 17 to April 6. A detailed description of the sprint and associated tasks can be found below.

Task ID	Title	Description	Start	End	Duration (days)	Asignee	Dependencies
2.1	Database Structure	Develop the exact database structure, including all fields and format.	~~3/10~~ 3/17	~~3/14~~ 3/21	5	Sravya	1.8
2.2	Course Class Framework	Create Java class for Course, implementing methods not dependent on other classes ~~(i.e. Course has list of Students, but this will not yet be implemented here due the dependency)~~	~~3/12~~ 3/19	~~3/15~~ 3/22	4	John	
2.3	Student lass Framework	Create Java class for Student, implementing methods not dependent on other classes.	~~3/14~~ 3/21	~~3/16~~ 3/23	3	Branch	
2.4	TextEntry Class Framework	Create Java class for TextEntry, implementing methods not dependent on other classes. Note this will be used for Posts in the class and group pages.	~~3/15~~ 3/22	~~3/18~~ 3/25	4	Branch	
2.5	Group Class Framework	Create Java class for Group, implementing methods not dependent on other classes.	~~3/18~~ 3/25	~~3/20~~ 3/27	3	Sravya	
2.6	Event Class Framework	Create Java class for Event, implementing methods not dependent on other classes.	~~3/19~~ 3/26	~~3/21~~ 3/28	3	John	
2.7	Finalize Classes	Implement remaining methods that were dependent on incomplete classes.	~~3/22~~ 3/29	~~3/26~~ 4/2	5	Branch	2.2-6
2.8	Database and Class Integration	Integrate class methods requiring database, using JSON to facilitate communication. Test reading and writing real data.	~~3/27~~ 4/3	~~3/30~~ 4/6	4	John	2.1 2.7

7.3 Sprint 3

In this sprint, we integrated the frontend User Interface with the backend database and Java objects. This included facilitating and verifying the communication between various components and the database. Additionally, we performed system testing and debugged the final issues before submitting the final application. For more testing information, see Section 7. This sprint ran from April 1 to April 19. A detailed description of the sprint and associated tasks can be found below.

Task ID	Title	Description	Start	End	Duration (days)	Asignee	Dependencies
3.1	User Authentication	Add authentication for users to login and signup on the app. Add methods to save usernames/passwords to database.	4/1	4/2	2	Branch	2.1
3.2	Profile Page Functionality	Display relevant data on the Profile page from the database in real time, including name, major, and graduation year. Add functionality to UI buttons (such as Edit Profile).	~~4/2~~ 4/7	~~4/4~~ 4/10	3	Sravya	1.3 2.3
3.3	Course Page Functionality	Display relevant data on the Course page from the database in realtime, such as posts, classmates, and list of study groups. Add UI button functionality (such as Create Post).	~~4/3~~ 4/9	~~4/6~~ 4/11	4	John	1.4 2.2
3.4	Add Course Page Functionality	Implement Add Course option such that a user can successfully add a course to their profile. Display user's current courses in side navigation bar.	~~4/4~~ 4/10	~~4/6~~ 4/12	3	Sravya	1.5 2.2
3.5	Groups Page Functionality	Display live data on the Groups page, including posts, members, and list of events. Add UI button functionality (such as Create Event).	~~4/6~~ 4/10	~~4/9~~ 4/12	4	John	1.6 2.5
3.6	Unit Testing	Test edge cases across application, including different user input conditions.	~~4/10~~ 4/13	~~4/14~~ 4/17	5	All	3.1-5
3.7	Final Debug	Debug final implementation of the application and make minor adjustments.	~~4/15~~ 4/17	~~4/19~~ 4/19	~~5~~ 3	All	3.6

7.4 Documentation

Our project documentation plan is includes the requirements, system architecture, and class diagram. Documentation began on April 10 and ended on April 17. A detailed description of this plan and associated tasks can be found below.

Task ID	Title	Description	Start	End	Duration (days)	Asignee	Dependencies
4.1	Requirements Document	Create necessary changes to requirements document, such as adding or removing requirements based on the final application.	~~3/15~~ 4/10	~~3/24~~ 4/14	~~10~~ 5	Sravya	
4.2	System Architecture	Edit the system architecture diagram to reflect final design decisions.	~~3/25~~ 4/15	~~3/28~~ 4/17	~~4~~ 3	John	1-3
4.3	Class Diagram	Make necessary changes to class diagram and add any additional classes or methods implemented.	~~3/29~~ 4/15	~~4/2~~ 4/17	~~5~~ 3	Branch	1-3
4.4	README	Create readme to inform users on how to build and run the application. Include a user's guide.	4/15	4/17	3	Sravya	1-3

8 Test Plan

Our testing plan is divided into unit, component, and system testing, lasting from April 13 to 19. For unit testing, we planned to perform mostly automated JUnit tests, but shifted to manual testing due to time constraints. For component testing, we performed mostly manual testing via inspection. For system testing, we performed a combination of requirements and use-case testing. We begin our testing with unit testing. Then, once all units comprising a given component passed, we began our component testing. Finally, once all components passed, we finished with the system test.

Section 7.1 to 7.3 cover unit, component, and system testing, respectively. The main components of our application include Login Component, Profile Component, Posts Component, Groups & Events Component, Course & Group Members Component, and Add Course Component.

8.1 Unit

Component	Test ID	Title	Description	Type	Start	End	Duration (days)	Asignee
C1	U1.1	Sign Up	Verify only valid email addresses and passwords are accepted.	~~JUnit~~ Manual	~~4/1~~ 4/13	~~4/1~~ 4/13	0.3	Branch
	U1.2	Authorization	Verify a provided email/password combination exists in database.	~~JUnit~~ Manual	~~4/1~~ 4/13	~~4/1~~ 4/13	0.3	Branch
	U1.3	Forgot Password	Verify either provided email address receives an email to reset password or user receives an alert if the email address is not recognized.	Manual	~~4/1~~ 4/13	~~4/1~~ 4/13	0.3	Branch
C2	U2.1	Edit Profile Picture	Verify provided image can be written/read to/from the database.	~~JUnit~~ Manual	~~4/2~~ 4/13	~~4/2~~ 4/13	0.3	Sravya
	U2.2	Edit Profile Text	Verify provided profile data can be written/read to/from the database. Check invalid inputs in all text fields.	~~JUnit~~ Manual	~~4/2~~ 4/13	~~4/2~~ 4/13	0.3	Sravya
C3	U3.1	Add/Remove Post	Verify posts can be added/removed to/from the database.	~~JUnit~~ Manual	~~4/2~~ 4/13	~~4/2~~ 4/13	0.3	Sravya
	U3.2	Add/Remove Comment	Verify comments can be add/removed to/from the correct post in the database.	~~JUnit~~ Manual	~~4/3~~ 4/13	~~4/3~~ 4/13	0.3	John
C4	U4.1	Add/Remove Group/Event	Verify that adding/removing a group/event is reflected in the database.	~~JUnit~~ Manual	~~4/3~~ 4/13	~~4/3~~ 4/13	0.3	John
	U4.2	Join/Leave Group/Event	Verify that joining/leaving a group/event is reflected in the database.	~~JUnit~~ Manual	~~4/3~~ 4/13	~~4/3~~ 4/13	0.3	John
C5	U5.1	Add/Remove Members	Verify that adding/removing members is reflected in the database.	~~JUnit~~ Manual	~~4/4~~ 4/14	~~4/4~~ 4/14	0.3	Branch

	U5.2	View Profile	Verify each member's name links to the correct profile page.	Manual	~~4/4~~ 4/14	~~4/4~~ 4/14	0.3	Branch
C6	U6.1	Save Course	Verify that an added course is reflected in the database.	~~JUnit~~ Manual	~~4/4~~ 4/14	~~4/4~~ 4/14	0.3	Branch
	U6.2	Search Course	Verify that a search keyword returns the appropriate data.	~~JUnit~~ Manual	~~4/5~~ 4/14	~~4/5~~ 4/14	0.3	Sravya
Other	U7.1	Invalid Input	Verify invalid inputs in all text fields are handled properly.	~~JUnit~~ Manual	~~4/5~~ 4/14	~~4/5~~ 4/14	0.3	Sravya
	~~U7.2~~	~~Boundary Values~~	~~Verify that all boundary values are respected, such as testing the maximum number of courses and groups that a user can add and the number of times a user can enter invalid logins.~~	~~JUnit~~	~~4/5~~ 4/14	~~4/5~~ 4/14	0.3	Sravya

8.2 Component

Component testing was performed after all unit tests passed.

Test ID	Title	Description	Type	Start	End	Duration (days)	Asignee
C1	Login	Verify upon a successful login, a user is taken to the profile view. Verify login information is stored in the database.	Manual	~~4/6~~ 4/15	~~4/8~~ 4/15	~~3~~ 1	John
C2	Profile	Verify edited profile information is reflected in the UI.	Manual	~~4/6~~ 4/15	~~4/8~~ 4/15	~~3~~ 1	Branch
C3	Post	Verify adding/removing posts and comments are reflected in the UI.	Manual	~~4/6~~ 4/15	~~4/8~~ 4/15	~~3~~ 1	Sravya
C4	Groups/Events	Verify that adding/removing groups/events is reflected in the UI. Verify joining/leaving groups/events is reflected in the UI.	Manual	~~4/9~~ 4/16	~~4/11~~ 4/16	~~3~~ 1	John
C5	Course/Group Members	Verify adding/removing members is reflected in the UI.	Manual	~~4/9~~ 4/16	~~4/11~~ 4/16	~~3~~ 1	Branch
C6	Add Course	Verify that added courses are reflected in the navigation bar and across the UI.	Manual	~~4/9~~ 4/16	~~4/11~~ 4/16	~~3~~ 1	Sravya

8.3 System

We performed end-to-end manual system testing to ensure full functionality of the application.

Test ID	Title	Description	Type	Start	End	Duration (days)	Asignee
S1	Requirements	Verify any remaining untested requirements, beginning with functional and advancing to non-functional requirements, time permitting.	Requirements	~~4/12~~ 4/17	~~4/15~~ 4/17	~~4~~ 1	All
S2	Use-case	Verify that all outlined use-cases are followed via manually testing the specific use-case within the application.	Use-case	~~4/12~~ 4/18	~~4/15~~ 4/18	~~4~~ 1	All
S3	Beta Testers	Allow several individuals to interact with the application and report any bugs.	User	~~4/15~~ 4/19	~~4/19~~ 4/19	~~4~~ 1	All
S4	Survey	From our beta users, feedback will be taken via a survey and utilized to make any remaining changes.	User	~~4/15~~ ~~4/19~~	~~4/19~~ ~~4/19~~	~~4~~ ~~1~~	~~All~~

9 Schedule

Below is a Gantt chart of the overall project plan, covering all sprints, documentation and tests. Note that Sprint 2 was delayed, which increased our workload towards the end of the semester.