PROJECT DOCUMENT

Created by: DMS^2

ABSTRACT

This project document contains all the necessary documents as specified by the course outline and IEEE. These documents speak to the creation of the Edu.Share software, ranging from why this project was chosen, to the source code used to create the software.

Group Members

Shemar Lundy
Dejeon Battick
Mark Anthony-Jones
Shemar Henry

Contents

1. Introduction

1.1 Proposal

The problem being solved

In Jamaica's current society there exists an evident disconnect between people with knowledge and people who seek knowledge. Take for example a student who is struggling with a subject area. This problem worsens when the individual is away from school and cannot seek help from the people around him/her due to their lack of knowledge on the subject matter. Another person has a growing interest in something that no one around them seem to understand or have an interest in. take another person who simply cannot learn, or grasp a concept the way a lecturer/teacher, or colleagues explain it and would appreciate another viewpoint on the topic. All these issues have one reoccurring theme, a gap between people who are actively seeking knowledge and people who have and can bring this knowledge across. In Jamaica's current society there doesn't exist an effective community whose sole purpose is to share the wealth of knowledge with people who seek it. Edu.Share seeks to combat this.

Why it's a good problem to solve?

The stated problem is good and worth solving due to two facts. Firstly, the Edu.Share team is made up of students and who have experienced this problem firsthand. Time and time again where the drive for knowledge is hindered or completely stopped due to the environment, the way the knowledge is being delivered and the inconvenience of pursuing this knowledge. Secondly, the issue goes under the radar especially in a developing country like Jamaica. There isn't many if any initiatives/programs being launched by the private or public sector to bridge the gap in the education system. It is important to note that lack of attention from these sectors doesn't mean the issue isn't worth pursuing, just means that Jamaica has yet to see the impact correcting this will have on the country as a whole.

Will your solution solve the problem?

Yes, the Edu.Share application will solve the issue because it is tackling the problem from a new point of view. Implementing this application in such a way that it feels familiar and comfortable to the user. The Edu.Share application will be created in such a way that it has a somewhat of a social media aspect to it, in turn pulling in more users. Also, it is intended to have the backing of a well-established educational institution therefore helping to the application the spread amongst the general population.

2. Project Management Documents

2.1 Project Charter

General Information

Project Title		Date
Edu. Share		09/11/2019
Project Manager	Phone	Email
Shemar R. Lundy	180 8671 2962	lundyshemar@yahoo.com
Executive Sponsor	Phone	Email
Dr Carl Beckford	NA	Carl.beckford@uwimona.edu.jm

Document Version	Updated Date
1.0	09/11/2019

Project Scope

Situation / Problem / Opportunity

The disconnect between persons within the education system is prevalent. With little to no technological community for these persons to communicate, creating a technological gap between persons with information and persons seeking information.

Project Goals

Creating a mobile application to bridge this technological gap. An application that creates an environment where individuals can educate and spread the wealth of knowledge easily. The mobile application platform would be suited best for this project due to wide availability of mobile phones capable of handling the intended application thus making access to the body of knowledge even easier.

In Scope

In Scope:

- Creating a reserve of scholarly documents that users will have access to.
- Allowing users to register as a tutor/student, then allowing student users to communicate with tutor users with the aid of a multi-media chat system to aid in the learning process.
- Creating a web view access to only credible websites (.org, .gov, .edu etc) where users can access information.

Objectives / Deliverables

- Creation of a database with the appropriate DBMS that allows users to read and write, within a certain degree, to the database.
- Creation of different types of user portals, namely the student user portal, the tutor portal and the publisher portal.
- Creation of a multi-media chat system.
- Creation of a web-search that guides users to only scholarly webpages.

Project Assumptions

- This project is unique in that no other software has all these features
- The high priority functional requirements will be finished before the deadline
- Implementation will be longer due to the skill level of the team

Risks and Dependencies

- Lack of resources or resource availability (time).
- Scope creep for additional 'predefined' reports.
- Inadequate feedback from requirements elicitation

Resource Requirements

- People: Executive Sponsor, programmers and project manager
- Time: The initial estimate for the project duration is approximately 12 weeks with an
- Other: availability of appropriate software.
- Budget: Project Manager billed at $450/hr worked, Database Developer, UI Developer and Server Developer billed at $300/hr worked

High-Level Milestones and Timeline

- Project Charter Approved: November 4, 2019
- Kickoff Meeting: November 7, 2019
- Identify and Document High Level Requirements: November 9, 2019
- Finalize Requirements: December 5, 2019
- Design and Development Completed: TBD
- Unit Testing Completed: TBD

Project Team Roles and Responsibilities		
Team Member	**Roles**	**Responsibilities**
Carl Beckford (PD)	Supervisor	The Supervisor will provide expert feedback to ensure the project deliverables are satisfactory for the stakeholders.
Shemar R. Lundy	Project Manager	The Project Manager develops and maintains the project plan (resourcing, implementation, work plans, etc.), monitors project progress, is responsible for all documentation and ensures projects are completed on time and within budget.
Mark-Anthony Jones	Database Developer	The database developer creates the database design and scheme, develops the database management system and maintains the database.
Dejeon Battick	UI Designer	The UI Designer creates the layouts for each platform (Android, iOS, Web) and develops other front-end functionality.
Shemar Henry	Backend Developer	The Backend Developers is responsible for programming a server API that connects the database to the server and the server to

	the client as well as providing server-side functions

Communication Strategy

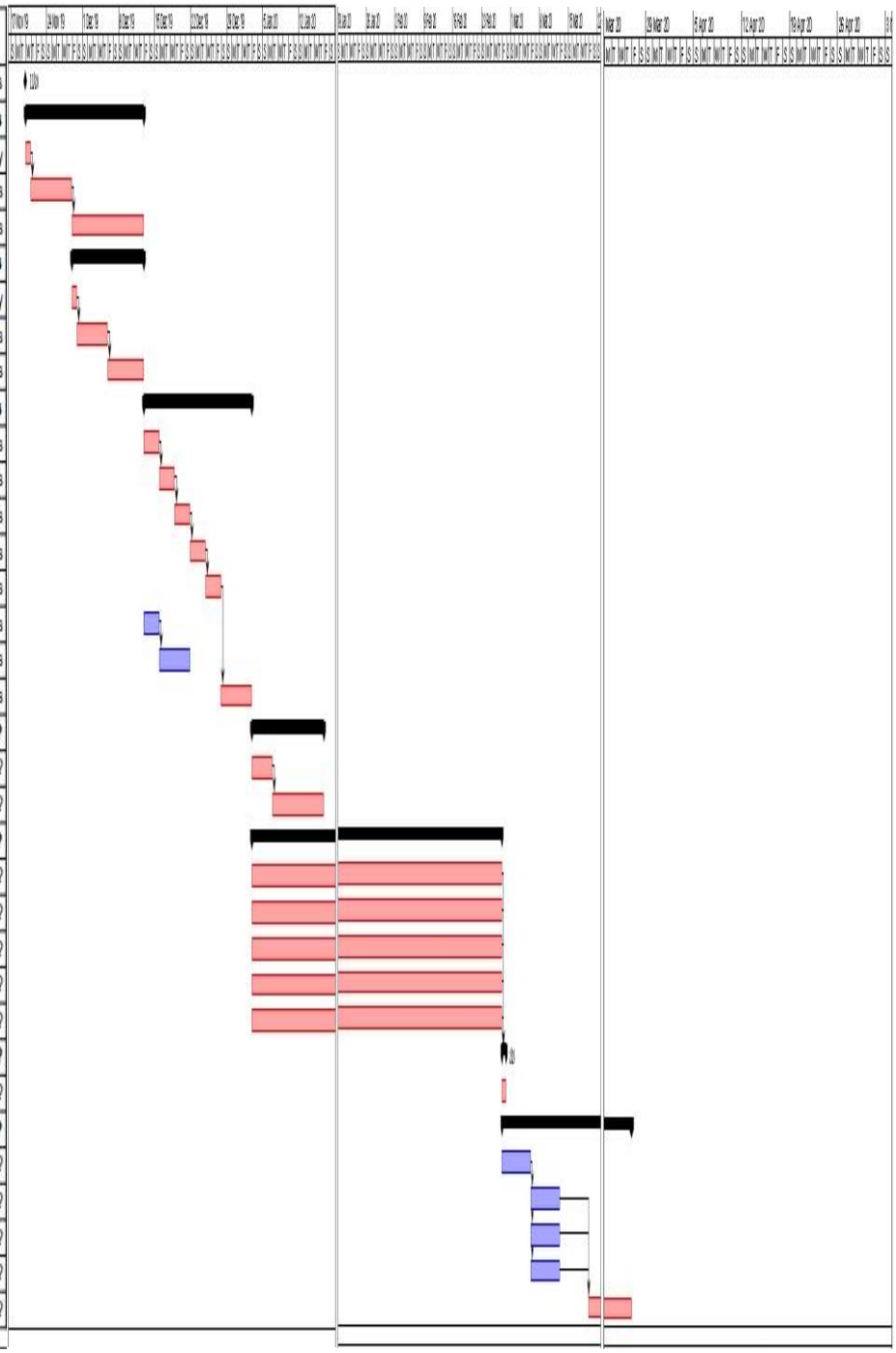
What?	Who?		When?	How?	
Description/Title	*Provider*	*Recipient*	*Frequency*	*Format*	*Medium/Distribution on Method*
Kickoff Meeting	Project Manager	Project Team	One Time Only	Meeting	Face to Face
Project Team Meetings	Project Manager	Project Team, supervisor	Weekly	Meeting	Face to Face Conference Call
Meeting Minutes	Project Manager	Distribution List	2 days after Meeting by COB	MS Word file on Shared Drive	Email
Action Items	Project Team	Project Team	TBD	MS Excel file on Shared Drive	Face to Face
Status Reports	Project Manager	Supervisor	Weekly / Bi-Weekly	PDF on Shared Drive	Email
Technical Design/Development Meetings	Project Manager Developers	Project Team	TBD	Meeting	Face to Face

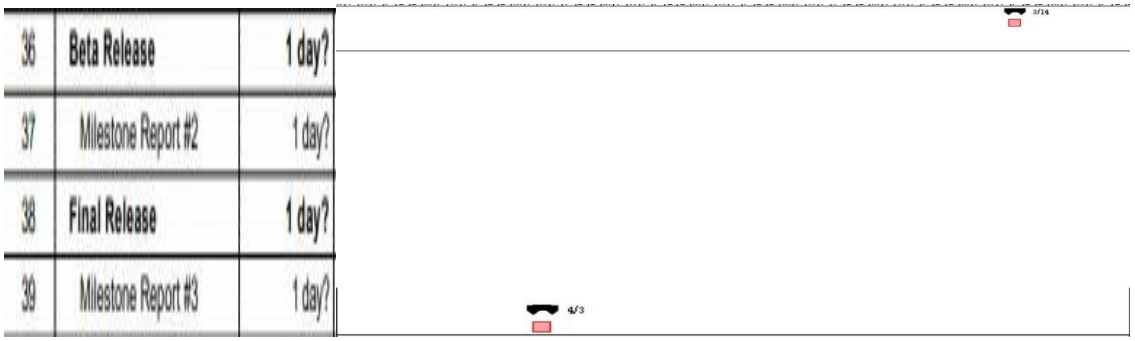
Charter will be reviewed and signed by the sponsors and manager					
Executive Sponsor		**ITSM Product Manager**		**Senior Manager, Business Intelligence**	
Reviewed By	Date	Reviewed By	Date	Reviewed By	Date

2.2 Project Plan

	Name	Duration	Start	Finish	Predecessors	Resource Names
1	Project Kickoff	0.042 days	11/19/19 9:00 PM	11/19/19 10:00 PM		
2	**Project Charter**	**23 days**	**11/19/19 10:00 PM**	**12/12/19 10:00 PM**		
3	Brainstorm	1 day	11/19/19 10:00 PM	11/20/19 10:00 PM		
4	Define Scope	8 days	11/20/19 10:00 PM	11/28/19 10:00 PM	3	
5	Produce Project Charter ...	14 days	11/28/19 10:00 PM	12/12/19 10:00 PM	4	
6	**Requirements Specific...**	**14 days**	**11/28/19 10:00 PM**	**12/12/19 10:00 PM**		
7	Requirements Ellicitation	1 day	11/28/19 10:00 PM	11/29/19 10:00 PM		
8	Requirements Document...	6 days	11/29/19 10:00 PM	12/5/19 10:00 PM	7	
9	Produce Software Requi...	7 days	12/5/19 10:00 PM	12/12/19 10:00 PM	8	
10	**Product Design and Arc...**	**21 days**	**12/12/19 10:00 PM**	**1/2/20 10:00 PM**		
11	Use Case Diagram	3 days	12/12/19 10:00 PM	12/15/19 10:00 PM		
12	Class Diagram	3 days	12/15/19 10:00 PM	12/18/19 10:00 PM	11	
13	Sequence Diagram	3 days	12/18/19 10:00 PM	12/21/19 10:00 PM	12	
14	State Diagram	3 days	12/21/19 10:00 PM	12/24/19 10:00 PM	13	
15	Deployment Diagram	3 days	12/24/19 10:00 PM	12/27/19 10:00 PM	14	
16	Entity Relationship Diagram	3 days	12/12/19 10:00 PM	12/15/19 10:00 PM		
17	Database Design Docum...	6 days	12/15/19 10:00 PM	12/21/19 10:00 PM	16	
18	Software Design Document	6 days	12/27/19 10:00 PM	1/2/20 10:00 PM	15	
19	**Prototype**	**14 days?**	**1/2/20 10:00 PM**	**1/16/20 10:00 PM**		
20	Produce Adobe XD Proto...	4 days?	1/2/20 10:00 PM	1/6/20 10:00 PM		
21	Develop Working Protot...	10 days?	1/6/20 10:00 PM	1/16/20 10:00 PM	20	
22	**Product Development**	**56 days?**	**1/2/20 10:00 PM**	**2/27/20 10:00 PM**		
23	Android UI Development	56 days?	1/2/20 10:00 PM	2/27/20 10:00 PM		
24	iOS UI Development	56 days?	1/2/20 10:00 PM	2/27/20 10:00 PM		
25	Web Front-End Develop...	56 days?	1/2/20 10:00 PM	2/27/20 10:00 PM		
26	Server-Side Development	56 days?	1/2/20 10:00 PM	2/27/20 10:00 PM		
27	Database Development	56 days?	1/2/20 10:00 PM	2/27/20 10:00 PM		
28	**Full Stack Application**	**1 day?**	**2/27/20 10:00 PM**	**2/28/20 10:00 PM**	**23;24;25;26;27**	
29	Milestone Report #1	1 day?	2/27/20 10:00 PM	2/28/20 10:00 PM		
30	**Software Testing**	**28 days?**	**2/27/20 10:00 PM**	**3/26/20 10:00 PM**		
31	Test Plan	7 days?	2/27/20 10:00 PM	3/5/20 10:00 PM		
32	Database Testing	7 days?	3/5/20 10:00 PM	3/12/20 10:00 PM	31	
33	Client Testing	7 days?	3/5/20 10:00 PM	3/12/20 10:00 PM	31	
34	Server Testing	7 days?	3/5/20 10:00 PM	3/12/20 10:00 PM	31	
35	Test Report	7 days?	3/19/20 10:00 PM	3/26/20 10:00 PM	34;33;32	
36	**Beta Release**	**1 day?**	**3/12/20 9:00 PM**	**3/13/20 9:00 PM**		
37	Milestone Report #2	1 day?	3/12/20 9:00 PM	3/13/20 9:00 PM		
38	**Final Release**	**1 day?**	**4/2/20 8:00 AM**	**4/3/20 8:00 AM**		
39	Milestone Report #3	1 day?	4/2/20 8:00 AM	4/3/20 8:00 AM		

	Name	Duration
1	Project Kickoff	0.042 days
2	**Project Charter**	**23 days**
3	Brainstorm	1 day
4	Define Scope	8 days
5	Produce Project Charter ...	14 days
6	**Requirements Specific...**	**14 days**
7	Requirements Ellicitation	1 day
8	Requirements Document...	6 days
9	Produce Software Requi...	7 days
10	**Product Design and Arc...**	**21 days**
11	Use Case Diagram	3 days
12	Class Diagram	3 days
13	Sequence Diagram	3 days
14	State Diagram	3 days
15	Deployment Diagram	3 days
16	Entity Relationship Diagram	3 days
17	Database Design Docum...	6 days
18	Software Design Document	6 days
19	**Prototype**	**14 days?**
20	Produce Adobe XD Proto...	4 days?
21	Develop Working Prototy...	10 days?
22	**Product Development**	**56 days?**
23	Android UI Development	56 days?
24	iOS UI Development	56 days?
25	Web Front-End Develop...	56 days?
26	Server-Side Development	56 days?
27	Database Development	56 days?
28	**Full Stack Application**	**1 day?**
29	Milestone Report #1	1 day?
30	**Software Testing**	**28 days?**
31	Test Plan	7 days?
32	Database Testing	7 days?
33	Client Testing	7 days?
34	Server Testing	7 days?
35	Test Report	7 days?




36 Beta Release 1 day?
37 Milestone Report #2 1 day?
38 Final Release 1 day?
39 Milestone Report #3 1 day?
3/14
4/3

Software Requirements Specification

for

Edu.Share

Version 2.0 approved

Prepared by

DMS^2

December 5, 2019

Revision History

Name	Date	Reason For Changes	Version
Dejeon Battick		Added to previously incomplete document	2.0

3.1 Introduction

3.1.1 Purpose

In aiming to creating a community to share the wealth of knowledge, special thought must be given to the functional requirements of the Edu.Share system.

The requirements must capture everything that the system wishes to accomplish while at the same time ensuring that the system is up to par with modern or similar applications.

Creating a system that allows for users to fetch, upload and critique files if so requested and allowed, it is important to ensure that major functional requirements capture these features. It is also important that other requirements help to enforce, maintain, or support those major requirements. Functional requirements must work together to ensure that the completed Edu.Share system does no less than what it was set out to accomplish, even in its most basic version.

Other important major functional requirements must speak to the chat feature of the application and managing the user and their information. The chat feature like the features listed earlier must be captured in major functional requirements as well as supported and enforced by minor functional requirements. Managing user information, are functional requirements that will along with minor requirements from other features, help to ensure that the Edu.Share system is just as good as or better than its counter parts.

The collection of requirements that are created for the Edu.Share system speaks volumes for how successful the system will be at accomplishing its goals and as such must be given special attention and done to the highest degree.

3.1.2 Document Conventions

Throughout this document you'll find that the main headings are typed in Times 18 bold. Sub-headings will be in Times 14 also bold while the content body is in Times 12. The font color throughout the document will be black and where needed bullet points or a numbered list will be used.

3.1.3 Intended Audience and Reading Suggestions

This document is intended for use by developers, testers, end-users and stalk holders. As the reader progresses through this document they will be informed about the Edu.Share software in a multitude of ways such as:

- purpose
- The scope
- References
- Functions
- User classes and characteristics
- Design and implementation constraints
- User documentation
- Assumptions and dependencies
- Hardware and Software interfaces
- System features

Reading the document, it is best for the reader regardless of his or her purpose to get a full understanding of why and what the application is therefore consulting the purpose, scope, user class and characteristics and functions. Beyond this point the documentation will begin to lean to the technical side for a few sections, making more geared towards developers and testers, speaking on functions, design and implementation constraints, assumptions and dependencies, hardware and software interfaces. The system features section will again allow all users to comfortably read about the application regardless of the audience they belong to.

3.1.4 Product Scope

Upon completion, Edu.Share should allow users to:

1. Communicate with each other
2. Submit documents to be peer reviewed
3. Submit documents to be added to the database
4. Query the database
5. Perform web searches
6. Make changes to their user profile

3.1.5 References

https://www.freecodecamp.org/news/what-is-an-api-in-english-please-b880a3214a82/

https://www.bmc.com/blogs/dbms-database-management-systems/

https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

3.2 Overall Description

3.2.1 Product Perspective

The Edu.Share application is a new, self-contained product that will be used across multiple platforms. The product will carry out its major functions by connecting the user to the Edu.Share server where from there they will be able to communicate with other users, query the database and carry out web searches. The figure below illustrates the Communication process between the end-users, the server, and the database using the internet.

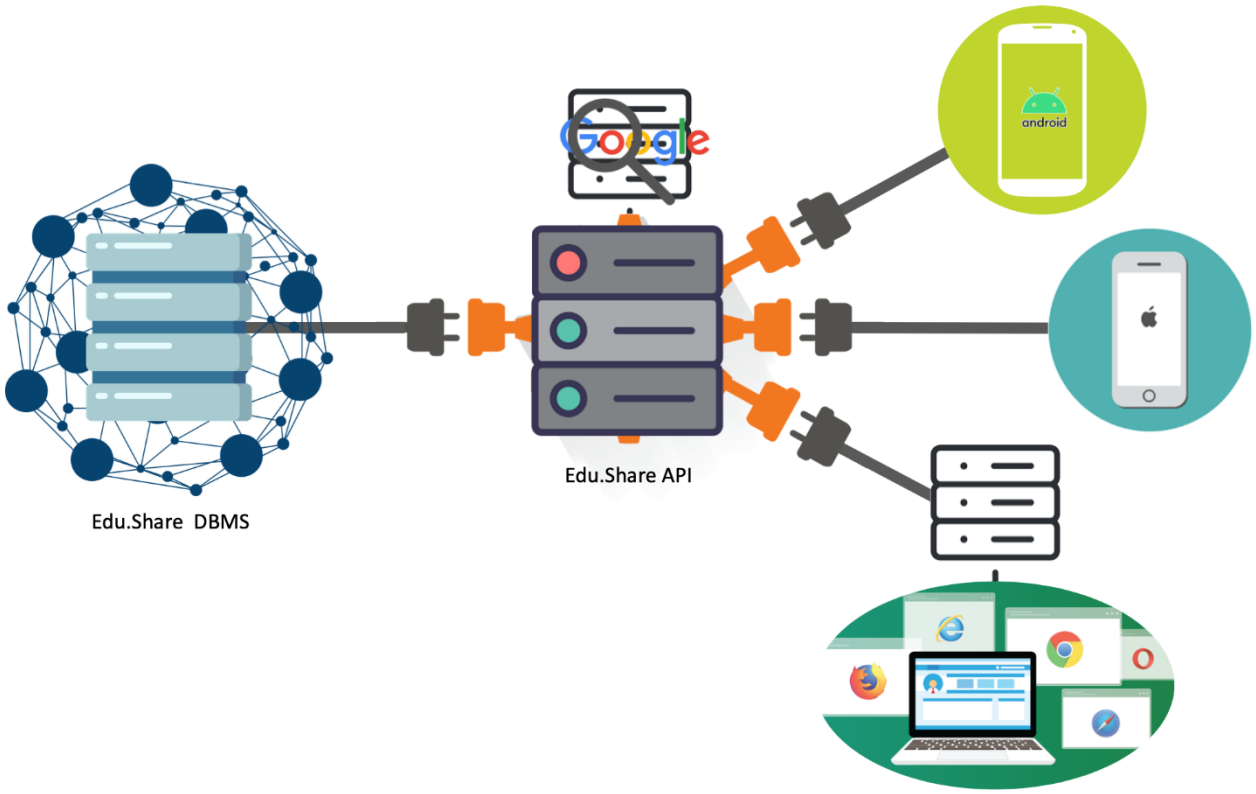


Fig 1.0 Edu.Share (See section 3 for more details)

3.2.2 Product Functions

- Create Profile
    - The user should be able set up their profile either as student, tutor or publisher.
- Search for document.
    - The user should be able to search for scholarly documents and filter their search for multimedia files.
- Read documents
    - The user should be able to read documents in app.
- Search for multimedia files.
- Perform web searches
    - User should be able to carry-out web searches from within the application.
    - Users should be able to search for multimedia files
- Play files
    - Users should be able to play multimedia files in the application
- Save to library
    - The user should be able to download documents/ multimedia files hence adding it to their library.
- Make bookmarks
    - The user should be able to save bookmarks.
- Find tutor

    - Students should be able to search for a tutor in a particular subject field in the chat.
- Save tutor
    - Students should be able to add preferred tutors to their favorites list
- Send messages
    - Users should be able to send text and multimedia messages.
- Update profile
    - Users should be able to update their profile information.
- Send chat request
    - Student users should be able to send chat requests to tutor and if need be cancel said request.
- Accept/decline chat request
    - Tutor user should be able to accept or decline chat requests
- Upload files
    - Users should be able to upload documents for peer review
    - Users should be able to upload reviewed documents
    - Users should be able to upload multimedia files
- Peer review documents
    - Peer reviewer should be able to view documents that are uploaded for review
- Send Notifications
    - The system should be able to send notifications to the user

3.2.3 User Classes and Characteristics

It is expected that the Edu.Share system will the following user classes:

- Student
    - Utilize multimedia chat to communicate with tutor users
    - Upload documents to be peer reviewed
    - Upload documents to the database after successful peer review
    - Specify who can see the document that they upload, public or a listed few.
    - Register as a peer reviewer if certain criteria are met
    - Edit user information
- Tutor
    - Utilize multimedia chat to communicate with student users
    - Upload documents to be peer reviewed
    - Upload documents to the database after successful peer review
    - Specify who can see the document that they upload; the public or a listed few.
    - Register as a peer reviewer if certain criteria are met
    - Edit user information

3.2.4 Operating Environment

The application will exist across a wide range of environments. The mobile version of the application will be operating as android and iOS applications

respectfully. There will also be a version of the application that can be accessed by a desktop or laptop computer.

3.2.5 Design and Implementation Constraints

Currently all developers are located in china and as such are hindered by certain political regulations. Certain tools such as google and its services, Zoom, just to name a few are not available and this will most definitely affect the rate at which work is done. There is also the added constraint due to time difference. This comes into place when weekly meetings are to be held via Zoom with the Edu.Share application's supervisor who is located in Jamaica.

3.2.6 User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

The Edu.Share software will be accompanied by a user manual along with in app help to help the user navigate the application.

3.2.7 Assumptions and Dependencies

Considering the nature of the Edu.Share project, the number one constraint when choosing tools was the experience as developers. There were no true restraint or restriction that superseded personal interest or desire to learn. However, it would be unwise to abuse this freedom, and instead familiar tools that would allow for asynchronous online collaboration was agreed on for efficiency sake.

These tools include:

- **GitHub** – online code hosting platform for version control and collaboration
- **Visual Studio Code** – a lightweight but powerful source code editor, includes support for debugging, embedded Git control and GitHub, syntax highlighting, intelligent code completion, snippets, and code refactoring.
- **Android Studio** – official integrated development environment (IDE) for Android application development, also has embedded Git control and GitHub
- **CleverCloud** – tool designed to make it easier to create, deploy, and run applications by using online containers
- **MySQL Workbench** – provides data modeling, SQL development, and comprehensive administration tools for server configuration, user administration, backup, and much more
- **SocketIO** – Javascript library that allows for real time bi-directional communication between web clients and servers.

The languages chosen for this project were:

- **Structured Query Language** – database management language for relational databases

- **Python** – the **flask** framework, a third-party Python library used for developing web applications; the **marshmallow** framework, library for converting complex datatypes, such as objects, to and from native Python and web friendly datatypes like JSON
- **Java** – the official language of Android development

The communication tools chosen for this project include the following:

- **Office 365** – line of subscription services offered by Microsoft as part of the Microsoft Office product line, allowing online collaboration on tools like Microsoft Word, Excel, PowerPoint and SharePoint (provided by UWI)
- **Trello** – a web-based Kanban-style collaborative project management tool that uses boards, lists, and cards to aid in organizing and prioritizing projects in a flexible, and rewarding way
- **Zoom** – a video communications (VOIP) tool, with an easy, reliable cloud platform for video and audio conferencing, chat, and webinars across mobile, desktop, and room systems
- **WeChat** – a Chinese multi-purpose messaging, social media and mobile payment app developed by Tencent (suitable since all project team members are currently located in China)
- **Outlook Email** – provided with Office 365, the main means of communication available between project supervisor and project team.

3.3 External Interface Requirements

3.3.1 User Interfaces

The login screen:

After opening the application for the first time, the user will be greeted with a login screen, displaying text fields for username/email and password respectively. Accompanying these fields will be a login button and a signup button. Clicking the login button will either grant the user access to the application if the given information is already in the database or alert the user of an error in what was being submitted. If the user doesn't have an account and clicks the sign-up button, then the user will be moved to the sign-up page where they can choose the account that they want to create. Giving the option to choose between a tutor or student account. After selecting the account, they wish to create, the user will then be moved to another screen that informs him\her about the features of the account they selected. The user can then confirm that this is the account they wish to create or opt to go back to the account selection page. If the user confirms the account selection, then they will be brought to the sign-up form. The user will also be able to sign up to be a peer reviewer by clicking on the "sign-up to be a peer reviewer". After clicking this button, the user will then be brought to another page where they are asked to submit the appropriate information to be acknowledged as a peer reviewer within the Edu.Share system. After completing this form and submitting the information the user will then be granted access to the application with the account they just created.

The home screen:

After a successful login or account creation, the user will then be brought to the home screen. On this screen the user will be presented with a search bar and quick search tiles. These tiles will allow the user to query the database quickly for scholarly documents. The tiles will have titles such as, chemistry, biology, forex etc. Aside from this, as mentioned before a search bar will be present allowing the user to query the database for a specific more narrowed search. After successfully carrying out a search the user will be brought to another window that displays everything in the database that relates to that search. The user may then scroll through the search results and will have the option of liking a document, therefore adding it to their favorite documents or clicking the read more option. Clicking this option will then bring the user to another screen where the abstract for the selected document. Also, the user will have the option to read the article now, download, add to favorites or go back. Clicking read now will bring the user to an in-application reader.

The chat screen:

The chat screen like the home screen will also have a search bar and quick search tiles. Unlike the home screen however these searches will query the database for tutors based on the subject they are proficient in. After performing this search, the user will be able to scroll through all the tutors registered to that particular subject area being searched. The user will also be able to like a tutor and request a chat with a particular tutor. After submitting a chat request the user will be brought to the in-app chat which in itself navigates between three windows; chat window,

chat request window and the favorites window. The chat window will have a list of all the chats the user had. Clicking on one of these chats will open a multimedia chat window, allowing the student and the tutor to communicate with texts, video, pictures, voice and or video calls. The chat request window for a student user will have all the chats that the user has requested, giving them the option to cancel a request if needed. The tutor chat request window will have the same functionality with the added feature of cancelling a request. The favorite window will list out all the tutors that the student user had previously liked.

The web search screen:

The web search window will allow the user to search the internet for credible and valid websites. The window will have a single search bar allowing the user to submit a search request, which will then move the user to a web view that will display the results of the search.

The user profile screen:

The user profile will consist of a list of functionalities that the user can accomplish in this window, namely; view chat, view favorites, view downloads, view bookmarks and edit profile. Clicking the view chat option will bring the user to the in-app chat window. The view favorites option will bring the user to a window with all their liked articles. The view download option will bring the user to a window listing all of the articles that the user has downloaded. Clicking the bookmark option will bring the user to a window that lists all the bookmarks that

the user has. Finally, the edit profile option will bring the user to a window that allows him/her to edit their profile information.

3.3.2 Hardware Interfaces

The hardware requirements for this solution are not complicated, only requiring access to the server network (which will be available through HTTP – see 3.4). The solution will be supported by Android Devices, Apple Mobile (iOS) Devices and any device with a Web Browser.

3.3.3 Software Interfaces

Edu.Share Database Management System:

The solution will have a dedicated centralized database system that will hold the general and operational information, working integrally with the server software interface but being independently manageable. This interface will also be the main to manage user control and privileges.

Edu.Share Server/Backend API:

The server or backend will be the main manager of solution protocols, providing an application programming interface (API) that the client/user interfaces will rely on for operation. It will delegate user privilege tasks and operations, connecting client to client and client to database.

Edu.Share Client/User Interfaces:

This solution will be available for Android, iOS and web platforms. These platforms will provide the interfaces that end users will use to access the server/backend API. They will be using well designed interactive tools that will allow for the defined system features to be accomplished, on communication with the server.

Google Services:

A very important feature of this solution is the search, which will be conducted using google API tools. Search results will be filtered by the Filter out the Foolishness (FotF) protocol, implemented by the Edu.Share server/backend.

3.3.4 Communications Interfaces

Edu.Share will use Hyper-Text Transfer Protocol (HTTP) to communicate between components, i.e. clients, server and database. The database will exist in a public domain where it is accessible to the server which will then allow client access via public REST API protocols. Security protocols will be installed at all nodes to ensure user and data safety and integrity.

3.4 System Features

3.4.1 Create profile

3.4.2 Search

3.4.2.1 Description and Priority

Upon creating a profile or logging into the application the user will be presented with the home page where they will be able to search for scholarly documents or multimedia files. The user can search by selecting a field or entering a keyword. After the search results are presented the user will then be able to filter their search by field, date released, alphabetical, document type and number of citing. The user swill then be able to read the documents or play the media files. Users can also use the web search feature to search official websites. This feature is of medium priority even though it is the main reason/ feature of the application the user can carry out other task within the application without using this feature.

3.4.2.2 Stimulus/Response Sequences

1. The user can select a field to search or type in a key word into the search bar
2. They system will present the user with the results of their search
3. The user can the choose to filter the search
4. The user can click on an item to read a document type file or play a media file

3.4.2.3 Functional Requirements

REQ-4: The user should be able search for a document

REQ-5: The user should be able to view documents

REQ-6: The user should be able to play media files

REQ-7: The user should be able to filter the search

REQ-8: The user should be able to perform web search – Users can search specialized websites (eg .edu .gov .org etc) and official websites ***

3.4.3 Chat

3.4.3.1 Description and Priority

The user will be able to access the chat features of the application be clicking on the chat icon on the navigation bar. From there students will be able to search for a tutor by specified fields or entering a user's username. They will also be able to view the tutor profile and request to chat with that user by sending a request message. The student can only continue to talk with that tutor if the tutor accepts the student's chat request. A tutor can view chat requests and accept or decline the request. If the tutor accepts the request, they can begin corresponding with the student sending text messages and other multimedia files.

3.4.3.2 Stimulus/Response Sequences

1. The user clicks on the chat icon in the navigation bar
2. The student search for a tutor to chat with
3. The student sends the tutor a chat request

4. The tutor accepts the chat request

5. The student and the tutor exchange message

3.4.3.3 Functional Requirements

REQ-8: A student should be able to send chat request

REQ-9: A student should be able to add tutors to their favorites list

REQ-10: Users should be able to send text and multimedia messages

REQ-11: A tutor should be able to accept a chat request

REQ12: A tutor should be able to decline a chat request

3.4.4 Upload files

3.4.4.1 Description and Priority

Users can upload files for two purposes, for peer review and to add to the database for public viewing. Files uploaded for peer review are private and only can be view by peer reviewers. The peer reviewer reviews these files and make recommendations then when the file is passed by two or more peer reviewer it is then fit for upload. The uploader will be notified, and they can make a decision whether to make it public or remain private, but it will be added to the database.

4.4.2 Stimulus/Response Sequences

1. The user from profile page clicks to upload file

2. If the file is peer reviewed the user can decide to make the file public or private
3. If the file has not been peer reviewed, then it is made private for only peer reviewers
4. Peer reviewer reviews the file
    i. If the file is passed by two or more reviewer, it is then eligible for upload
        1. The system notifies the uploader
        2. The uploader can decide to upload the file private or public
    ii. If the file fails its review
        1. The uploader is notified of the status of the file

3.4.4.2 Functional Requirements

REQ-13: The user should be able upload documents for peer review

REQ-14: The user should be able upload documents to add to the database

REQ-15: The user should be able to upload multimedia files

REQ-16: The user should be able to peer review documents

REQ-17: The system should be able to send notifications to users

3.5 Other Nonfunctional Requirements

3.5.1 Performance Requirements

During the process of searching via keyword, results should be listed in less than 1 second. When a message is sent in the chat the receiver should receive the message in less than 1 second after the sender sends the message.

3.5.2 Safety Requirements

To ensure the safety of the software the following requirements are put into place.

- Files to be added to the database can be no larger than the maximum specified size.
- A security scan will be performed on all files before adding them to the database to ensure that the file in question is not a malicious one.

3.5.3 Security Requirements

To ensure user and data security the following requirements are put into place.

- User credentials must be checked against the database to ensure user validity.
- Attacks such as SQL injection will be prevented by the use of good coding standards.

3.5.4 Software Quality Attributes

- Adaptability: the software should guarantee the same level of performance regardless of the operation environment (android, IOS, desktop or laptop computer). This can be done by using the appropriate language to allow the application to seamlessly transition from platform to platform

- Availability: the user should be able to search, upload documents, create profile and chat as long the proper conditions are met. This will be accomplished using a server that connects the user to the database and through which the user can access the internet. The user should be able to carry out other functions when offline line such as read view saved files and edit profile.

- Reliability: while using the software the user should be able to rely on the database to produce credible and reliable files. To ensure reliability of the software all files that are added to the database must first be successfully peer reviewed.

- Robustness: the software should be able to handle many users accessing the server at any given time in carrying out a variety of activities. It should also be able to handle errors without crashing the software. To accomplish this the software will utilize the necessary error checks and utilize a server.

- Usability: the software should be user friendly to the wide variety of users that will access it. To ensure this the user interface will not be overly crowded or complex and use neutrally toned colors.

- Testability: the software should be testable and to accomplish this, functional requirements should be clear and unambiguous and themselves testable.

3.5.5 Business Rules

- Student
    - Peer review documents given that they have the appropriate certifications
    - Add files to the database
    - Request and chat with tutors given that the request was accepted by said tutor
- Tutor
    - Peer review documents given that they have the appropriate certifications
    - Add files to the database
    - Chat with student users after the chat request has been accepted
    - Decline chat requests
- Developers
    - Make changes to the application functionalities
    - Make changes to the database

3.6 Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

3.7 Appendix A: Glossary

API: Application Programming Interface. According to freecodecamp.org, an API "is the part of the server that receives requests and sends responses."

DBMS: Database Management system according to BMC.com, "is the Technology solution used to optimize and manage the storing and retrieval of data from databases."

HTTP: According to developer.mozilla.org, HyperText Transfer Protocol is "a protocol which allows for the fetching of resources such as HTML documents."

FotF: Filter out the Foolishness is a protocol created by Edu.Share developers to filter the resources that are received.

HTML: HyperText Markup Language according to developer.mozilla.org, is "code that is used to structure the webpage and its content."

4. Software Design Document

(DMS^2)
(Edu.Share)

Names:
Shemar R. Lundy
Mark Anthony-Jones
Shemar Henry
Dejeon Battick

Date: (01/09/2020)

4.1 INTRODUCTION

4.1.1 Purpose

The Edu.Share application due to its major goal of creating a community to share knowledge amongst users it is assumed that the system will; one, be accessing large amounts of data at any given time, and two, be handling many different users at any given time. With this in mind, when designing the application, special though must be given on allowing multiple users to access and use the application at the same time without affect performance. Queries must also be efficiently written to handle the many different user requests in a timely manner again ensure optimal performance.

The system must also written in such a way that it carries out its functional requirements efficiently while at the same time has a certain level of robustness. Achieving these two criteria, the arrangement of classes and database tables must be done with keen vision allowing for classes to communicate efficiently and not waste time carrying out non-relevant functions or repeating code. Classes must be designed to ensure that functions are carried out quickly, functions are carried out within its appropriate classes and all coding standards are upheld. When designing the database for the Edu.Share system, being mindful of data integrity and security is of the utmost importance. A properly normalized database will aid in efforts to maintain data integrity.

The user interface for the Edu.Share system is also another important aspect of the system's design. Due to the wide target audience, the interface must be designed in such a way that is

appealing to a wide age group and backgrounds. The design must flow seamlessly allowing the user to easily carry out activities without tediously navigating the system windows. Ensure that the user has a good time while using the system is of key importance.

The communication between the different parts of the system is also integral in the way in which the system performance and thus the success of the system. Ensuring that all components of the Edu.Share system is the optimal choice and connected for optimal performance will ensure that the system is working at its highest capacity. The overall design of the system from classes to API to subsystems is very important in determining not just if the system meets its goals but how well these goals are met.

4.1.2 Scope

Upon completion, Edu.Share should allow users to:

1. Communicate with each other
2. Submit documents to be peer reviewed
3. Submit documents to be added to the database
4. Query the database
5. Perform web searches
6. Make changes to their user profile

4.1.3 Overview

The software design document consists of several diagrams each allowing the reader to view the system to be built from a unique angle. Class diagrams, sequence diagrams, component diagrams, ER diagrams, use case diagrams and state diagrams can all be found here. Along with these diagrams the document consists of several tables that; one, shows the link between requirements, components and data structures, and two, defines the data associated with the ER diagram. The document also reiterates the purpose, scope and functionalities of the system to be created.

4.1.4 Reference Material

https://www.google.com/url?sa=t&source=web&rct=j&url=https://sovannarith.files.wordpress.com/2012/07/sdd_template.pdf&ved=2ahUKEwi-iZLLzpnnAhWOhJ4KHRlECigQFjAWegQIBhAB&usg=AOvVaw0ZbVIX9ceZ-OwXkHGvkhCT

https://www.ably.io/concepts/socketio

4.1.5 Definitions and Acronyms

- DB: database
- TCP/IP: transmission control protocol/internet protocol
- ER diagram: entity-relationship diagram

4.2 SYSTEM OVERVIEW

The system to be built should allow users to communicate, search for, read and or play scholarly files, chat with tutor users through various forms of messaging, upload documents to be peer

reviewed and added to the database and peer review documents themselves. Functionalities also include but are not limited to, editing user profile and saving and downloading documents. The Edu.Share system is intended to be used to aid in individual learning and sharing knowledge among users. Creating an environment or community geared towards sharing knowledge. With this in mind, along with the varying age groups of the intended users, the system's physical designed is done in such a way that is familiar to users and at the same time not overly complicated or too simple. In turn satisfying as many users as possible. The architectural design allows for the system to handle large sums of data without its performance being affected, a crucial aspect in the lifespan of such an application.

SYSTEM ARCHITECTURE

4.2.1 Architectural Design

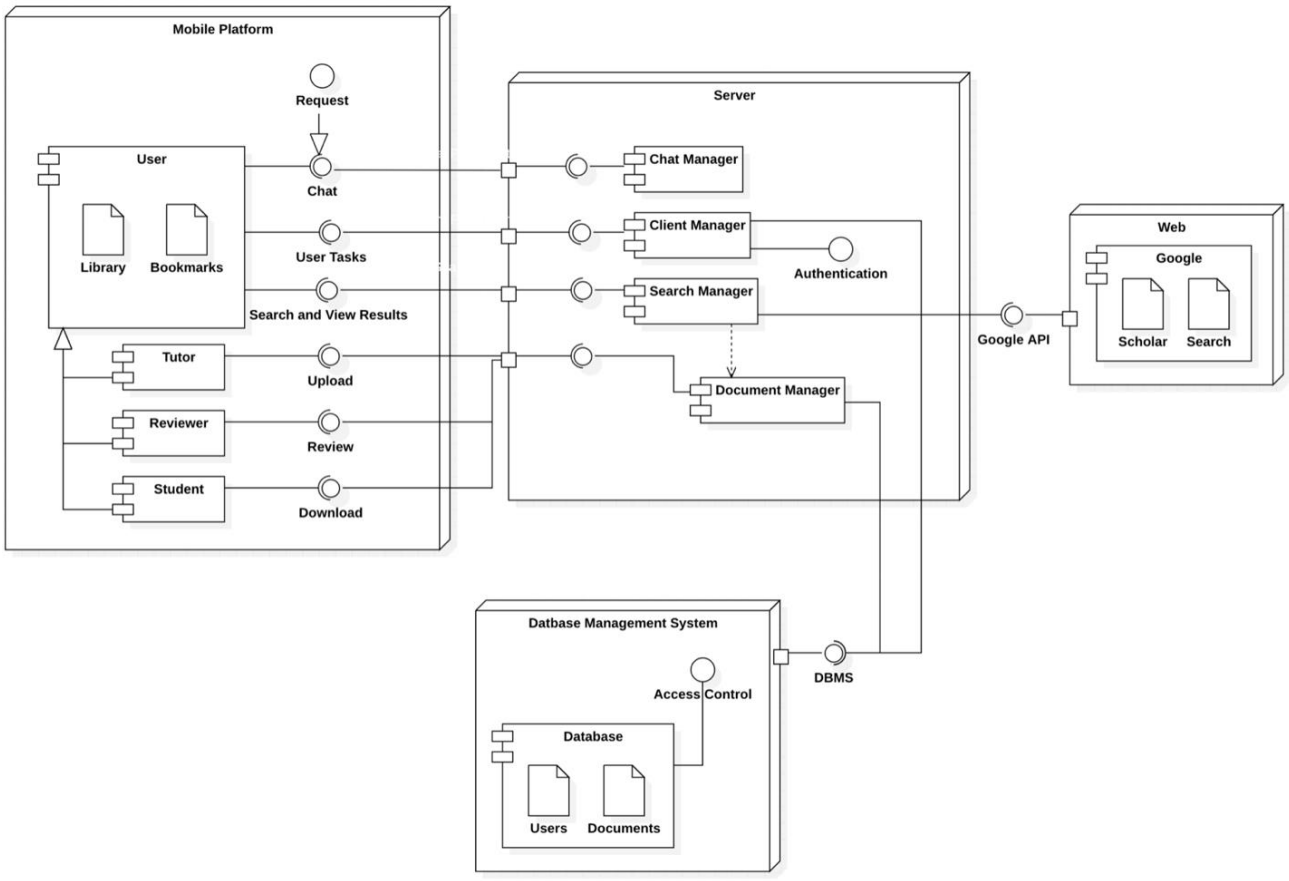


Figure 1: Deployment Diagram

The above diagram illustrates the nodes and components of the Edu.Share system and how they would interact with each other. All observed node interactions follow Transmission Control Protocol or the Internet Protocol (TCP/IP). The Server node will have the access to resources and services achieved by Edu.Share. Access to these services are provided by a mobile node/client, which through TCP/IP communicates with the server through the provided ports and interfaces of the server. The Database Management System node is the manager of resources, which supplements the server through TCP/IP. A web-based Google

API will be implemented as well, also communicated via TCP/IP. This is portrayed in the Web node, specifically referring the Google resources to be used for this project.

Class Diagram

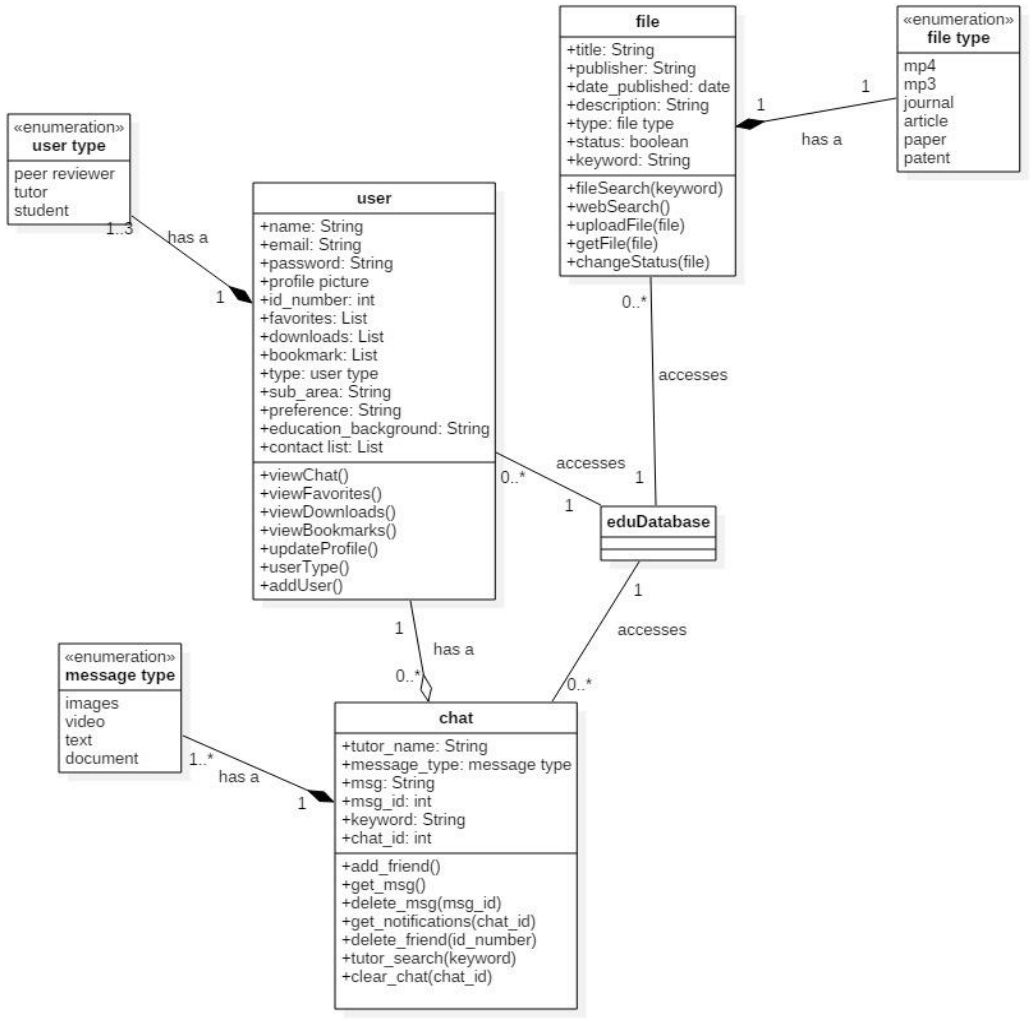


Figure 2. Class diagram

The above diagram illustrates the classes of the Edu.Share system and how they would interact would each other. As seen above there is a user class that captures the user information and stores it to the database to be used later. The user class calls on an enumeration user type to allow the user to specify if he or she would like to be a tutor, student or peer reviewer. The type function will then base on the type selected, if any will display the necessary additional fields. The file class captures all the necessary information

about the files to be stored. This class utilizes the enumeration ‘file type’ to capture the type of file that is being stored. The chat class will handle the sending and retrieving of messages between users and the message type will handle the type of message that are sent. Here it also allows the user to make alterations to their chat such as deleting messages and or deleting the entire chat. This class also allows the user to delete a person from their contact list, search for a tutor and send a chat request.

Use Case

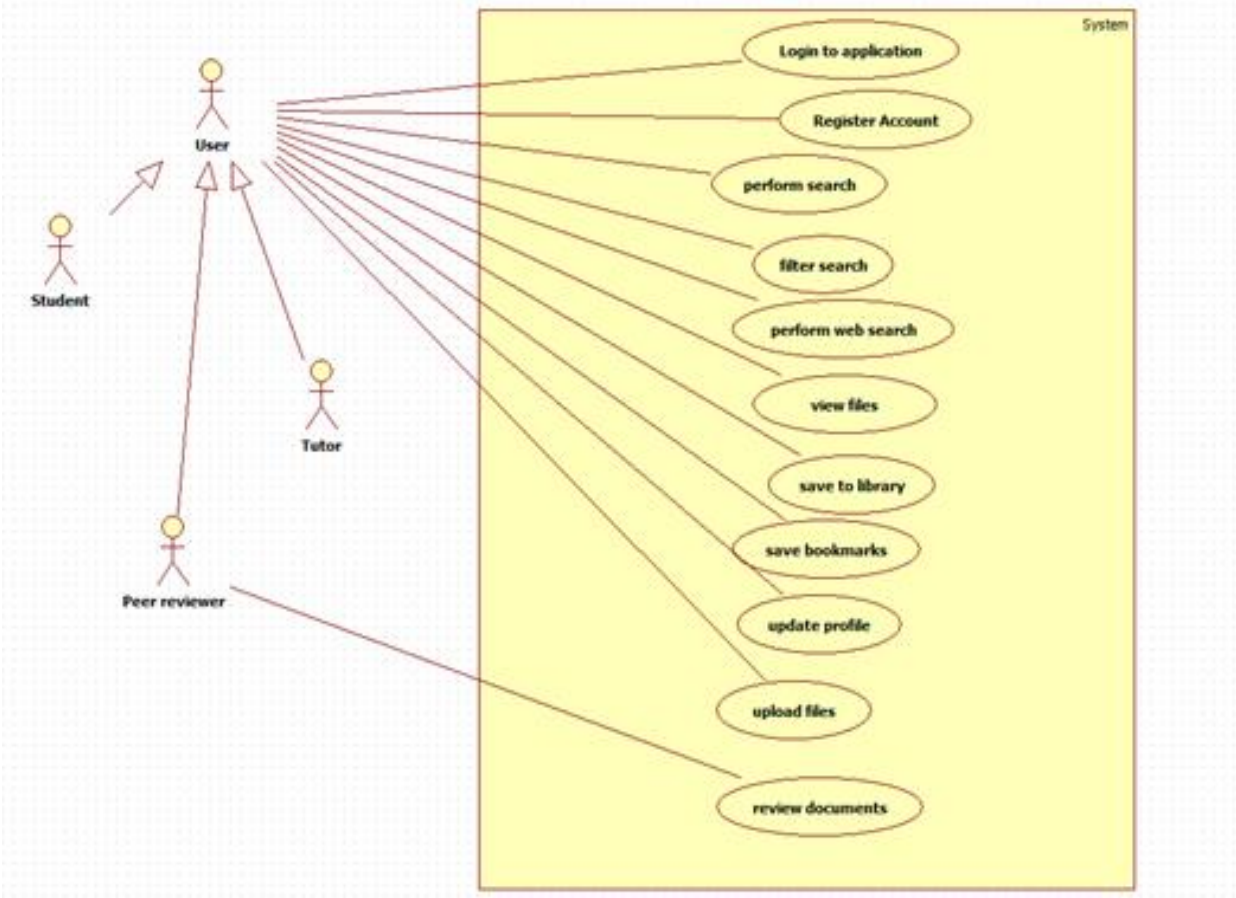


Figure 3: Use case diagram

The Actors include a student, tutor and peer reviewer who share a general association to users. Diagram one is a depiction of the system's use cases.

Use Case Name: Register account

Description: A user can register a new account as a student or a tutor

User priority: High

Actors: Users

Pre-conditions: Download and install the application

Trigger: Selects register new user

Main flow:

1. The user opens the application
2. The user chooses to register new account
3. The user selects to register as a student or tutor
4. The system prompts the user to enter their information
5. User enter required information
6. The system adds the new user to the database and send the user to the login page

Alternative Course:

1. If the user selects to register as a peer reviewer
2. The system prompts the user to enter additional information
3. The system confirms the user's information
4. System grants the user a peer reviewer status

Post-condition: A new user is successfully created, and the user is sent to the login page

Use Case Name: Login to Application

Description: A registered user can login to the application have access to its features

User priority: High

Actors: Users

Pre-conditions: user is registered

Trigger: The user opens the application

Main flow:

1. The user enters their username
2. User enter their password
3. System confirm username and password match
4. If they match the user is successfully logged in
5. System send the user to the home page

Alternative Course:

4.1. If the username and password do not match the user can try again

4.2. The system will prompt the user to try again or to reset their password

Post-condition: The user is sent to the home page of the application

Use Case Name: Perform search

Description: The users of the application will be able to search the database for files. This can be done by clicking on the search bar and typing key words or by searching by study area fields.

User priority: high

Actors: User

Pre-conditions: the user has logged into the system

Trigger: Click on the search bar or click on an area of study field

Main flow:

1. The user clicks on the search bar
2. The user types in key words
3. The system provides a list of files that match the keywords that was entered

Alternative Course:

1.2. The user clicks on a subject area field

1.3. The system provides a list of files that are associated to that field of study

Post-condition: A list of files that corresponds to the keyword or the field of study

Use Case Name: Perform web search

Description: The user should be able to perform searches for credible websites for the information that they are seeking

User priority: high

Actors: Users

Pre-conditions: The user has successfully logged into the system

Trigger: click on the wed search page

Main flow:

1. The user clicks on the search bar
2. The user types in a keyword and search
3. The system provides a list of websites that the user can visit

Alternative Course:

1. The user clicks on a field of study
2. The system provides a list of websites that match that subject area

Post-condition: The system provides a list of websites that match the user's search

Use Case Name: filter search

Description: The user should be able to filter their search results to get more specific results to suit their needs. They can filter to specify the ty of file, year publish, area of study etc.

User priority: Medium

Actors: Users

Pre-conditions: perform a search

Trigger: search results are listed

Main flow:

1. The user clicks on the filter icon
2. The user specifies the conditions they wish to filter the search by
3. The system provides a list of the specified filtered results

Post-condition: a list of files specific to the user's filter conditions

Use Case Name: View files

Description: The user should be able to open and view files of different format

User priority: high

Actors: users

Pre-conditions:

Trigger: user open a file

Main flow:

1. The user clicks to open the file
2. The file is open in a suitable viewer

Post-condition: The file is open and made available to the user

Use Case Name: Save to library

Description: The user should be able to save files to their library

User priority: medium

Actors: users

Pre-conditions: view files

Trigger:

Main flow:

1. The user views a file
2. The user chooses to save the file

3. The system adds the file to the user's library

Post-condition: The file is added to the user's library

Use Case Name: Save bookmark

Description: The user should be able to save or mark the last time they have read a document.

User priority: medium

Actors: user

Pre-conditions: The user open a file

Trigger: The user clicks bookmark

Main flow:

1. The user opens a file
2. The user bookmark where they last left off
3. The system saves the mark to the user's library

Post-condition: The document is added to the user library

Use Case Name: update profile

Description: the user should be able to make changes to their profile

User priority: medium

Actors: users

Pre-conditions: the user have an account

Trigger: the user clicks edit profile

Main flow:

1. The user goes on profile page
2. The user selects edit profile
3. The user makes changes to their profile
4. The user confirms the changes
5. The system saves the user changes

Post-condition: The user changes are saved

Use Case Name: Review documents

Description: A user who registered as a peer reviewer can review documents that have been submitted to be reviewed,

User priority: medium

Actors: Peer reviewer

Pre-conditions: the user can view an uploaded document to be reviewed

Trigger: the user opens the document

Main flow:

1. The user opens the document
2. The user reviews the document

3. The user mark that they have reviewed the document

Post-condition: The document has been marked as reviewed

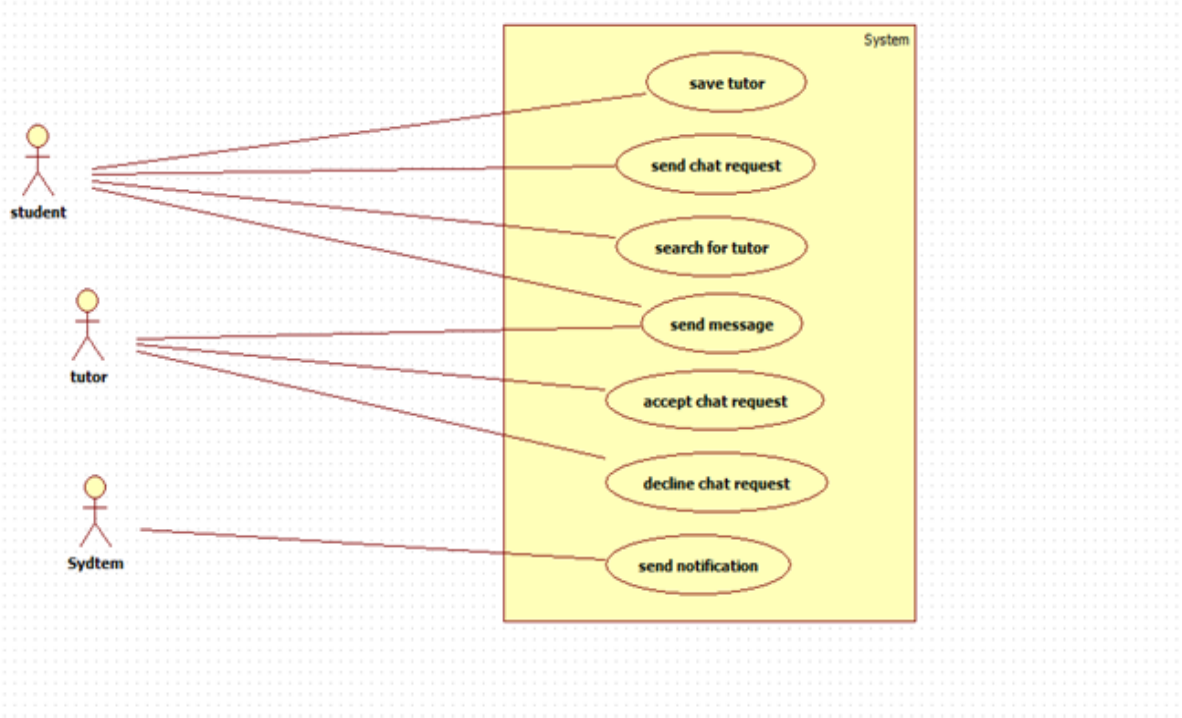


Figure 4: Use case diagram

The above diagram shows that the:

- A student user should be able to save a chat request, send chat request, send a message and search for a tutor
- A tutor user should be able to send a message, accept chat requests, and decline chat request.
- The system should be able to send notifications

Use Case Name: Save tutor

Description: A student can save a tutor of their liking to their chat

User priority: medium

Actors: Student

Pre-conditions: A student view a tutor profile

Trigger:

Main flow:

1. The user views a tutor's profile
2. The user chose to add the user to their favourites
3. The system saves the tutor to the user's favourites

Alternative Course:

Post-condition: The tutor is added to the user's favourites

Use Case Name: Send chat request

Description: The Student send the tutor a request to chat

User priority: high

Actors: Student

Pre-conditions: A student view a tutor profile

Trigger: The student views the tutor profile

Main flow:

1. A student views the tutor's profile
2. A student sends a request chat message to the tutor

Post-condition: The tutor gets the request message

Use Case Name: Search for tutor

Description: A student can search for a tutor by their name, email or they can search by a field of study.

User priority: medium

Actors: Student

Pre-conditions: A student open the chat feature

Trigger: click to search for tutor

Main flow:

1. A student clicks on the search bar
2. A student enters the tutor name or email, or search be field of study
3. A system shows the results of the search

Post-condition: The list of search results are listed

Use Case Name: Send message

Description: users can send multimedia messages to each other after the tutor accepts the chat request from the student.

User priority: high

Actors: users

Pre-conditions: the tutor accept the chat request from a student

Main flow:

1. The tutor accepts a chat request from the student
2. The student sends a message to the tutor
3. The tutor responds

Post-condition: users send messages to each other

Use Case Name: Accept chat request

Description: The

User priority: high

Actors: tutor

Pre-conditions: a student send a chat request to the tutor

Trigger:

Main flow:

1. The tutor views the chat request from the student
2. The tutor choses to accept the chat request from the student

Post-condition: The student and tutor can now send messages to each other

Use Case Name: Decline chat request

Description: The tutor declines a request to chat from the student

User priority: medium

Actors: Tutor

Pre-conditions: a student send a chat request to the tutor

Trigger:

Main flow:

1. The tutor views the chat request from the student
2. The tutor choses to decline the chat request

Post-condition: the student can no longer message the tutor

Use Case Name: Send notification

Description: The system should be able to send alerts to the user in the case of a new message. The system will also send notification messages to the users.

User priority: high

Actors: System

Trigger: And alert or event

Main flow:

1. The users sent a message
2. The system notifies the other user that they have a message
3. A notification alert comes in

Post-condition: An alert message is sent to the user

4.2.3 Decomposition Description

Sequence Diagram

The sequence diagrams below illustrate the 4 main high priority functions of the application. These include registering for the application, searching for files, the use of the chat feature to communicate with tutors and the peer review process.

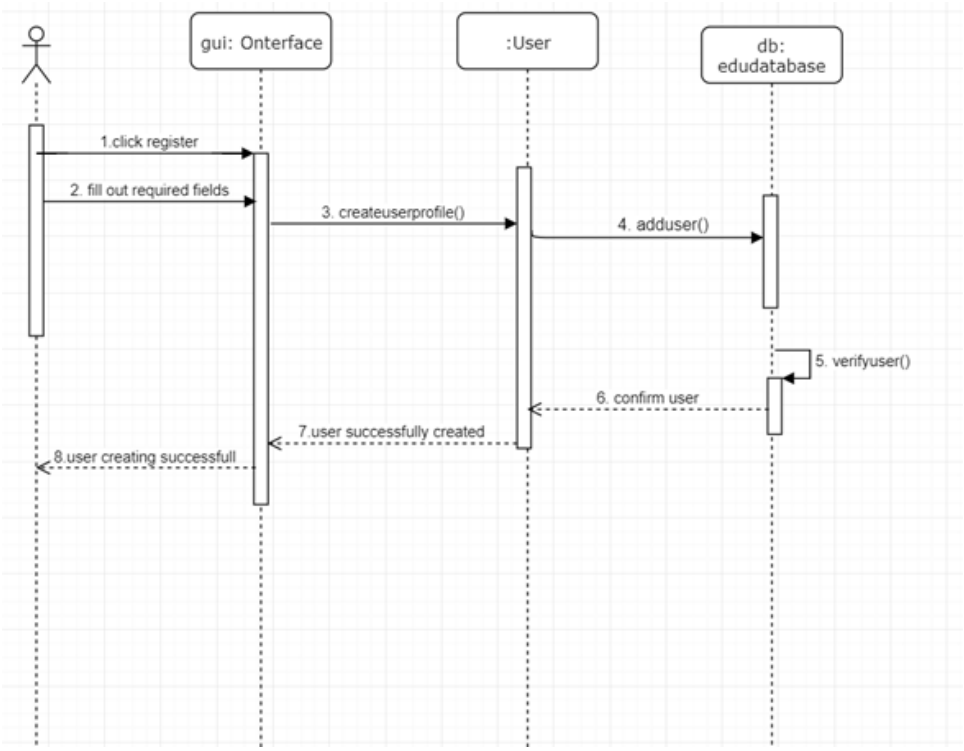


Figure 5 sequence diagram

This sequence diagram illustrates the registration process. Users can register as a student or a tutor, they can also choose to be a peer reviewer and have the privilege of reviewing documents that have been uploaded for review. When the user opens the application, they

will be on the login page, they then select to go to registration. From there they will be prompted to fill out the fields then click confirm to continue. The system will perform a check to ensure the all the fields are filled out correctly and if confirmed the app will then prompt the user to login with their credentials.

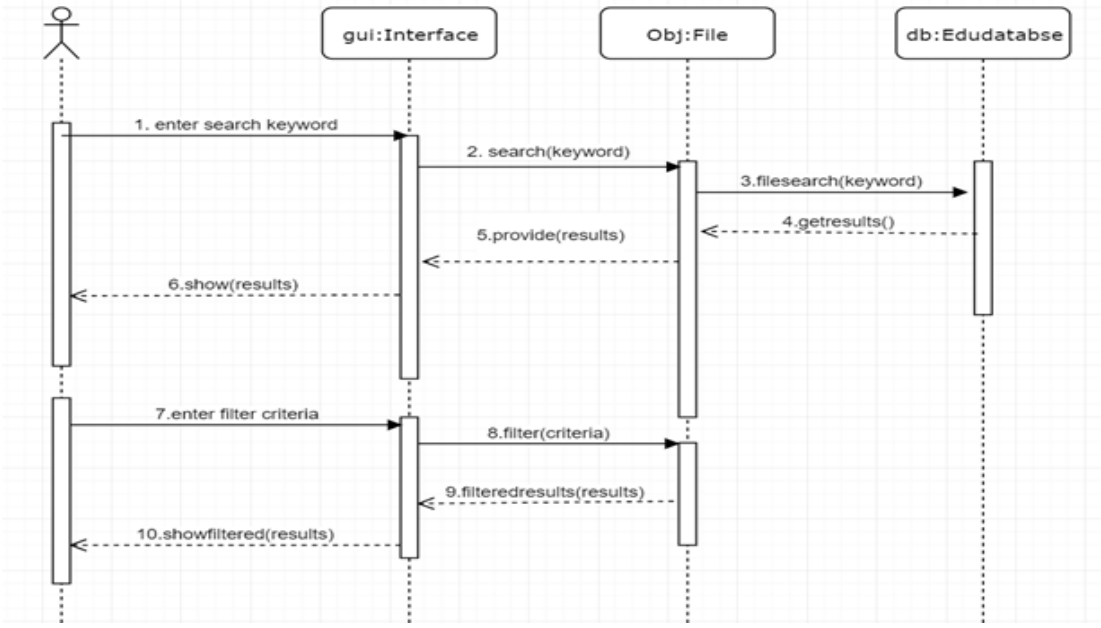


Figure 6 sequence diagram

This sequence diagram illustrates the process to search for files and documents. After logging in to the system the user will be sent to the search page which is the home page. The user can then search by typing in a keyword or clicking on one of the subject area fields that are present on the screen. The system will the show a list of results to the user the user can then filter those results to their specific needs.

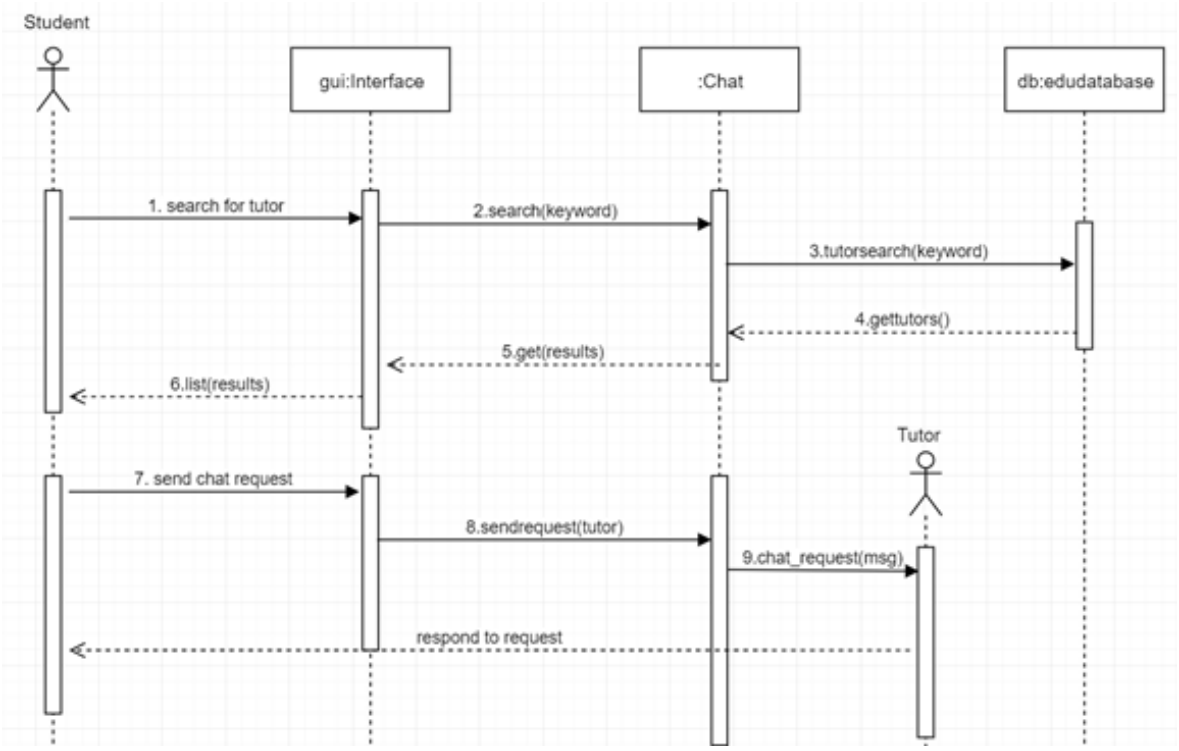


Figure 7 sequence diagram

This sequence diagram illustrates using the chat feature. A student can search for a tutor by typing in their username or they can search by subject area. After they have found the user, they are looking for proceed to sending the tutor a request to chat message which the tutor can then accepts or decline. If the tutor accepts then the line of communication is open, and they can message each other freely within the application.

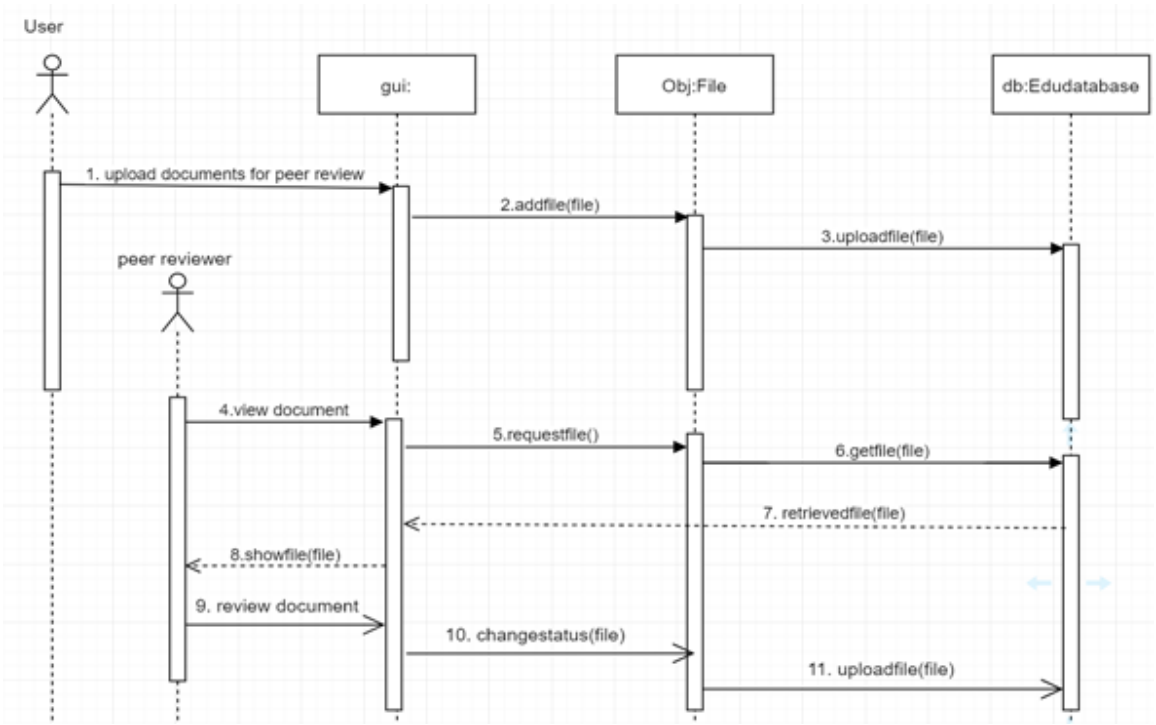


Figure 8 sequence diagram

This sequence diagram illustrates the peer review process. Only users who have registered to be a peer reviewer can review documents. A user can upload a document to be reviewed and it will have a special tag to it showing that it is to be reviewed. Peer reviewers can then view these documents and review it. When the document has been reviewed and pass the review criteria the status of the document is then changed. The uploader will then be notified, they can then do as they please with their document.

4.3 DATA DESIGN

4.3.1 Data Description

The data being transferred through the components of the system are stored and accessed by both a relational database management system and a non-relational database management system. The relational DBMS, MySQL, is responsible for storing login information, user information, information on user's roles and accessible services as well as storing peer reviews of critiqued documents posted in the application by users. The non-relational database, MongoDB, will be responsible for storing.

4.3.2 Data Dictionary and State chart diagrams

The data dictionary below is arranged alphabetically by the "Attribute Name" field.

Primary Key	Attribute Name	Required	Type	Field Length	Default Values	Notes
N	chat_content	Yes	Text	Unlimited	n/a	A text file containing an archived chat
Y	chat_id	Yes	Integer	20	n/a	Automatically generated ID for every

						instance of a chat opened
N	contact_list	No	Text	Unlimited	n/a	A list of user's contacts
N	document_authors	Yes	Text	255	n/a	A string containing the name(s) of authors of a document
N	document_file	Yes	Blob	n/a	n/a	A blob containing a file
Y	document_id	Yes	Integer	20	n/a	Automatically generated ID for every instance of a document saved
N	document_reviews	Yes	Text	255	n/a	A block of text describing a document
N	document_title	Yes	Text	30	n/a	Title of a document

N	Email	Yes	Text	30	n/a	User's email addressed used for login
Y	login_id	Yes	Integer	10	n/a	Automatically generated ID for every instance of a user
N	media_authors	Yes	Text	255	n/a	A string containing the name(s) of the authors for a file
N	media_file	Yes	Blob	n/a	n/a	A blob containing a media file
Y	media_id	Yes	Integer	20	n/a	Automatically generated ID for every media file uploaded
N	media_title	Yes	Text	30	n/a	A media file's title

N	Password	Yes	Text	255	n/a	A hashed version of a user's password
N	request_description	No	Text	255	n/a	A string of text a user sends along with a request
Y	request_id	Yes	Integer	255	n/a	Automatically generated integer to uniquely identify each request
N	request_title	Yes	Text	30	n/a	A string "<user_name> has sent you a chat request"
N	review_description	Yes	Text	255	n/a	The content of a review
Y	review_id	Yes	Integer	30	n/a	A unique integer for identifying each review

N	review_rating	Yes	Integer	1	n/a	An integer ranging between 0-5 denoting a rating
N	review_signature	Yes	Blob	n/a	n/a	An image file containing a peer reviewer's signature
N	review_title	Yes	Text	30	n/a	The title of a review
Y	role_id	Yes	Integer	10	n/a	Unique identifier for each role
N	role_name	Yes	Text	30	n/a	A string containing the name of a role
Y	service_id	Yes	Integer	10	n/a	Unique identifier for each service
N	service_name	Yes	Text	30	n/a	A string of containing the

						name of a service
Y	user_id	Yes	Integer	30	n/a	A user's unique ID
N	user_name	Yes	Text	50	n/a	A user's first and last name

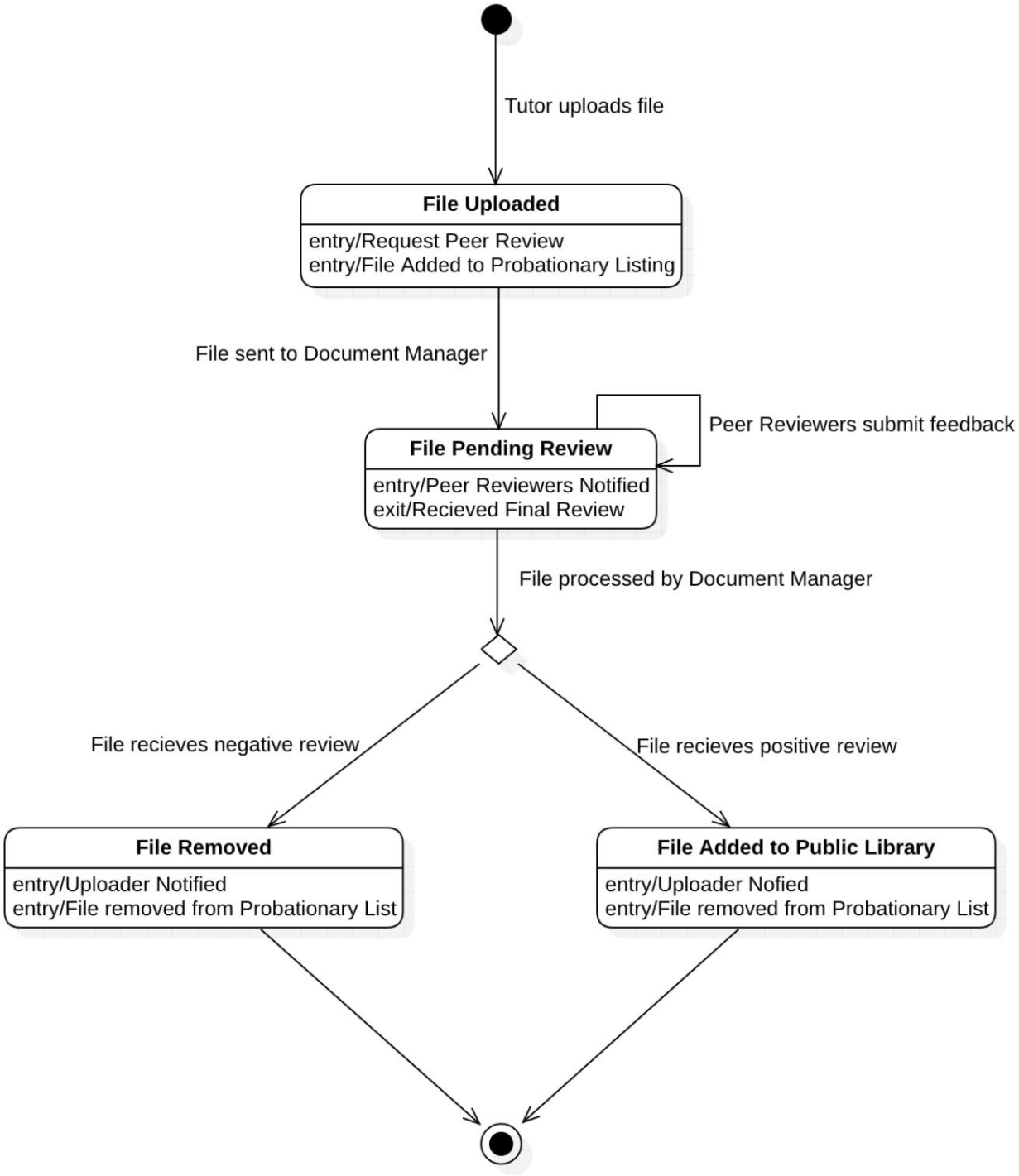


Figure 10. Edu.Share State Chart

The above diagram depicts uploading and reviewing a file. The diagram shows where a file is uploaded by a user, in this case a tutor user, and is sent to the document manager. The file is then peer reviewed by the authorized individuals and then depending on the feedback that is received the file will then be added or removed from the database.

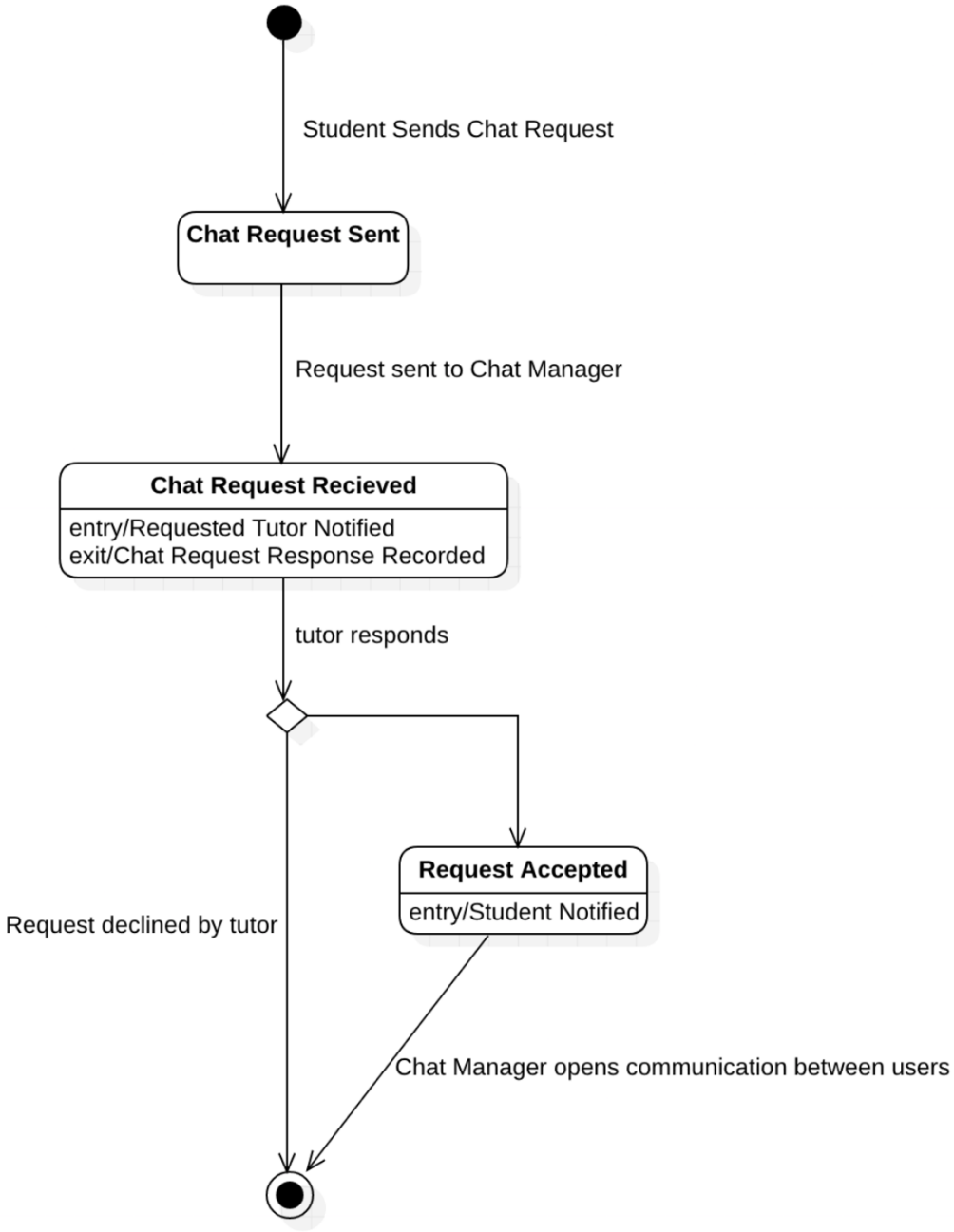


Figure 11. Edu.Share State Chart

The above diagram depicts a chat request. To start off the student send the chat request at which point the request is sent to the chat request manager. The request is then received

by the tutor notifying them of the request. The tutor then responds by either accepting the request or denying it.

4.3.3 Design Rationale

The resulting diagrams follow the identification of the tools and frameworks found most suitable for this project's requirements. The foundation for this architecture is the client/server architecture, since Edu.Share will feature many users interacting with the services and each other. The foundation was then built upon with the tools identified, which were categorized forming the nodes for the Edu.Share deployment diagram.

The chosen tools had their own features, like in the instance of the mobile client (Android-Java), resulting in the class diagram to compensate for a more Object-Oriented Design approach.

4.4 COMPONENT DESIGN

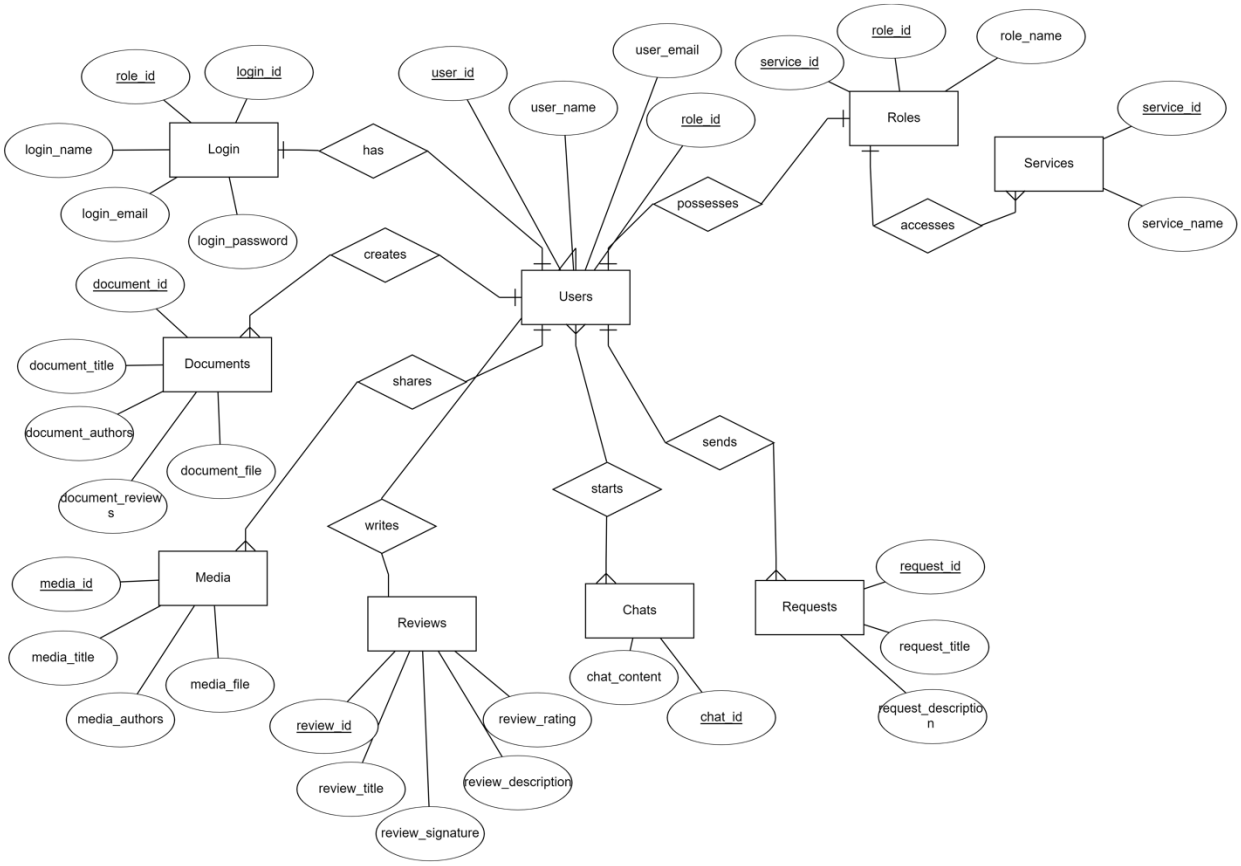


Figure 9. Edu.Share Database Management System Entity Relationship Diagram

Each user is given a login and a role upon registering. Roles are given services – operations which a user of a role is allowed access to. Examples of services include writing peer reviews, requesting chats to start instant messaging and sharing media and documents. A user may send as many requests as they wish and participate in as many chats as they choose to. A user is also able to share many documents and media files. This system implements a Role-Based Access Control (RBAC) architecture to restrict users from access features they are not qualified to use.

4.5 HUMAN INTERFACE DESIGN

4.5.1 Overview of User Interface

The login screen:

After opening the application for the first time, the user will be greeted with a login screen, displaying text fields for username/email and password respectively. Accompanying these fields will be a login button and a signup button. Clicking the login button will either grant the user access to the application if the given information is already in the database or alert the user of an error in what was being submitted. If the user doesn't have an account and clicks the sign-up button, then the user will be moved to the sign-up page. On this page the user will first be asked if they are interested in being a tutor or peer reviewer, if they so desire, they may select either options, one or none. If the user chooses to add one or both responsibilities to their account, then additional information will be requested. Information such as preferences, proof of educational background and the subject area(s) interested in. If the user doesn't want to add any of these responsibilities than they will simply continue to fill out the standard form. Importantly a user may 'verify' their account by uploading proof of their educational background. This however is optional and may be done at a later date. After completing this form and submitting the information the user will then be granted access to the application with the account they just created.

The home screen:

After a successful login or account creation, the user will then be brought to the home screen. On this screen the user will be presented with a search bar and quick search tiles. These tiles will

allow the user to query the database quickly for scholarly documents. The tiles will have titles such as, chemistry, biology, forex etc. Aside from this, as mentioned before a search bar will be present allowing the user to query the database for a specific more narrowed search. After successfully carrying out a search the user will be brought to another window that displays everything in the database that relates to that search. The user may then scroll through the search results and will have the option of liking a document, therefore adding it to their favourite documents or clicking the read more option. Clicking this option will then bring the user to another screen where the abstract for the selected document. Also, the user will have the option to read the article now, download, add to favourites or go back. Clicking read now will bring the user to an in-application reader.

The chat screen:

The chat screen like the home screen will also have a search bar and quick search tiles. Unlike the home screen however these searches will query the database for tutors based on the subject they are proficient in. After performing this search, the user will be able to scroll through all the tutors registered to that particular subject area being searched. The user will also be able to like a tutor and request a chat with a particular tutor. After submitting a chat request the user will be brought to the in-app chat which in itself navigates between three windows; chat window, chat request window and the favourites window. The chat window will have a list of all the chats the user had. Clicking on one of these chats will open a multimedia chat window, allowing the student and the tutor to communicate with texts, video, pictures, voice and or video calls. The chat request window for a student user will have all the chats that the user has requested, giving them the option to cancel a request if needed. The tutor chat request window will have the same

functionality with the added feature of cancelling a request. The favourite window will list out all the tutors that the student user had previously liked.

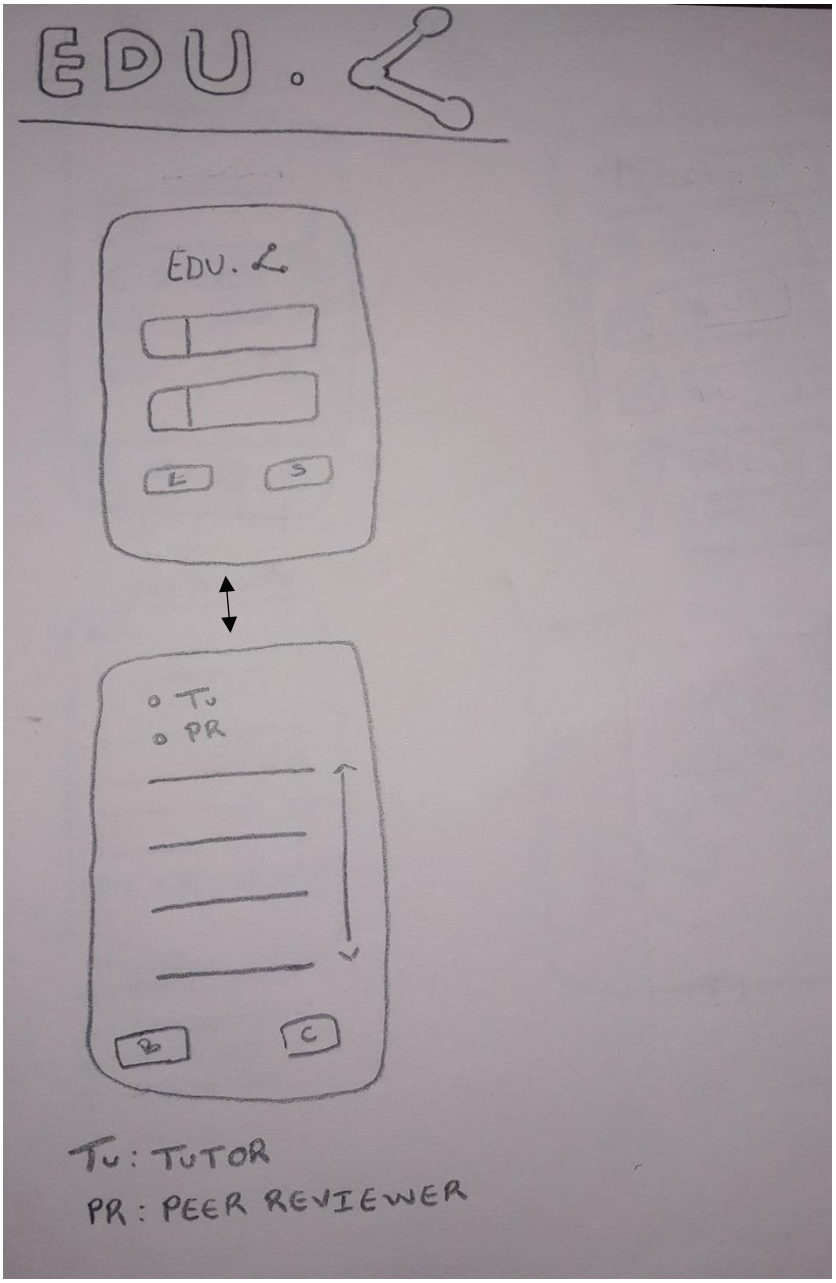
The web search screen:

The web search window will allow the user to search the internet for credible and valid websites. The window will have a single search bar allowing the user to submit a search request, which will then move the user to a web view that will display the results of the search.

The user profile screen:

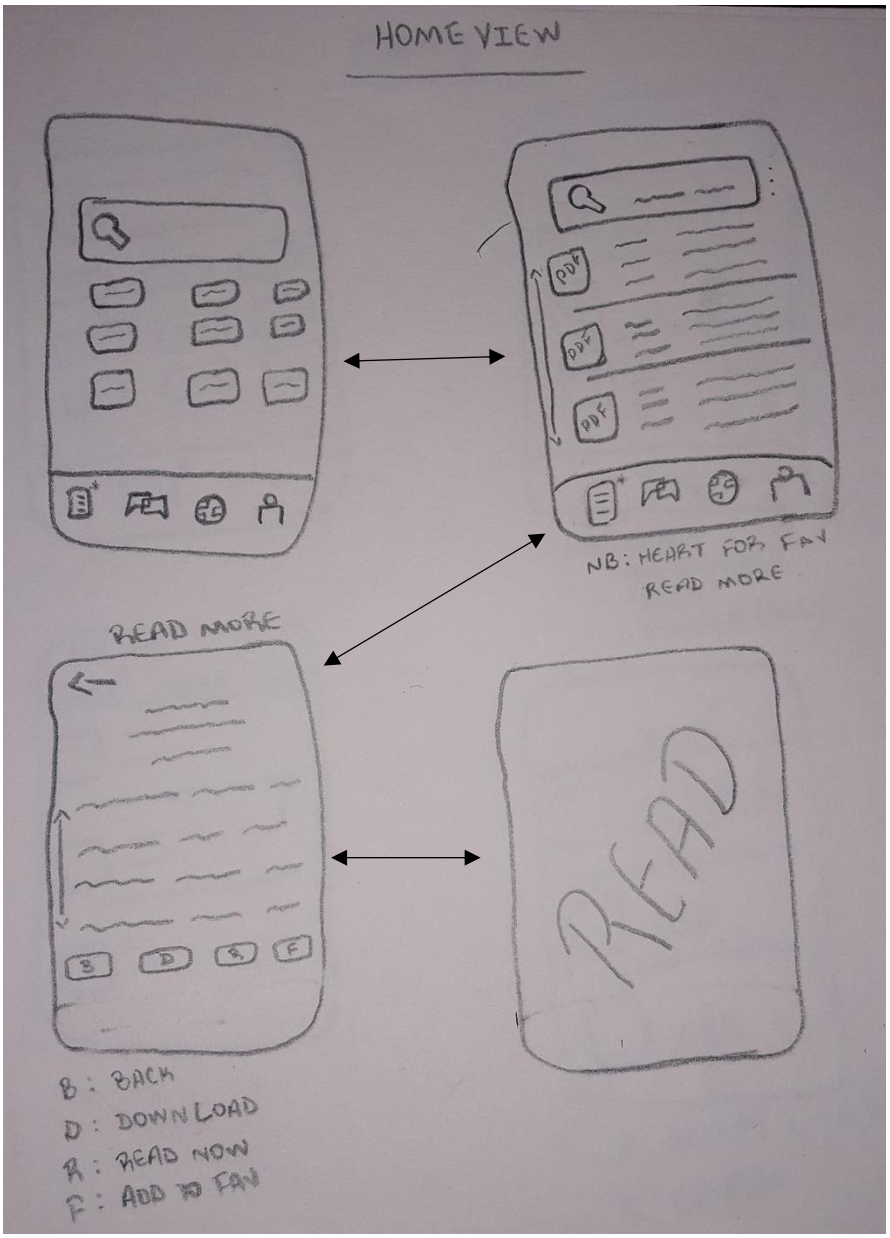
The user profile will consist of a list of functionalities that the user can accomplish in this window, namely; view chat, view favourites, view downloads, view bookmarks, edit profile and add a file. Clicking the view chat option will bring the user to the in-app chat window. The view favourites option will bring the user to a window with all their liked articles. The view download option will bring the user to a window listing all of the articles that the user has downloaded. Clicking the bookmark option will bring the user to a window that lists all the bookmarks that the user has. The edit profile option will bring the user to a window that allows him/her to edit their profile information. This page is where a user may verify their account as mentioned earlier or add additional responsibilities to their account. If the user's account is verified they will be given an additional option on the user profile screen and that is to upload a file. This allows a verified user to upload a file to be peer reviewed and added to the database if its peer review was successful.

4.5.2 Screen Images



Key:

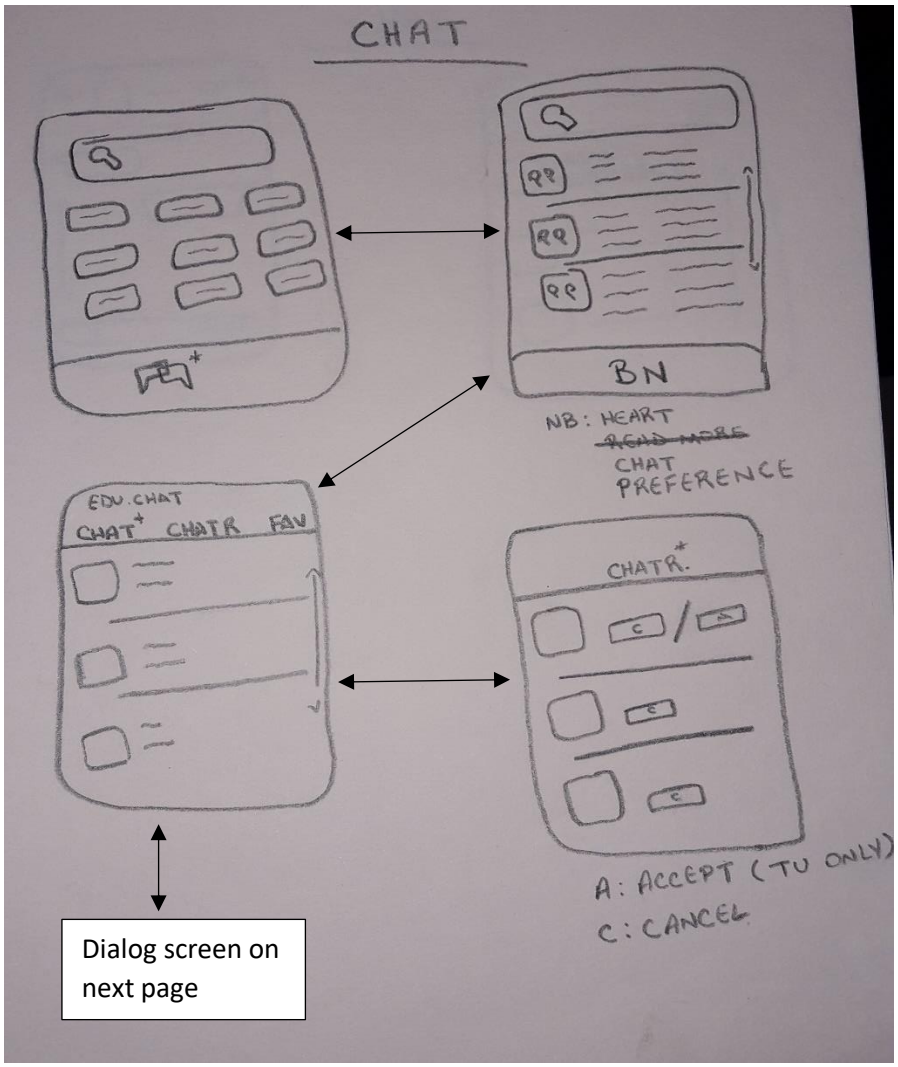
- L: login button
- S: signup button
- Tu: tutor
- PR: peer reviewer
- B: back
- C: continue
- Hand drawn vertical arrow shows that the screen should be scrollable



Key:

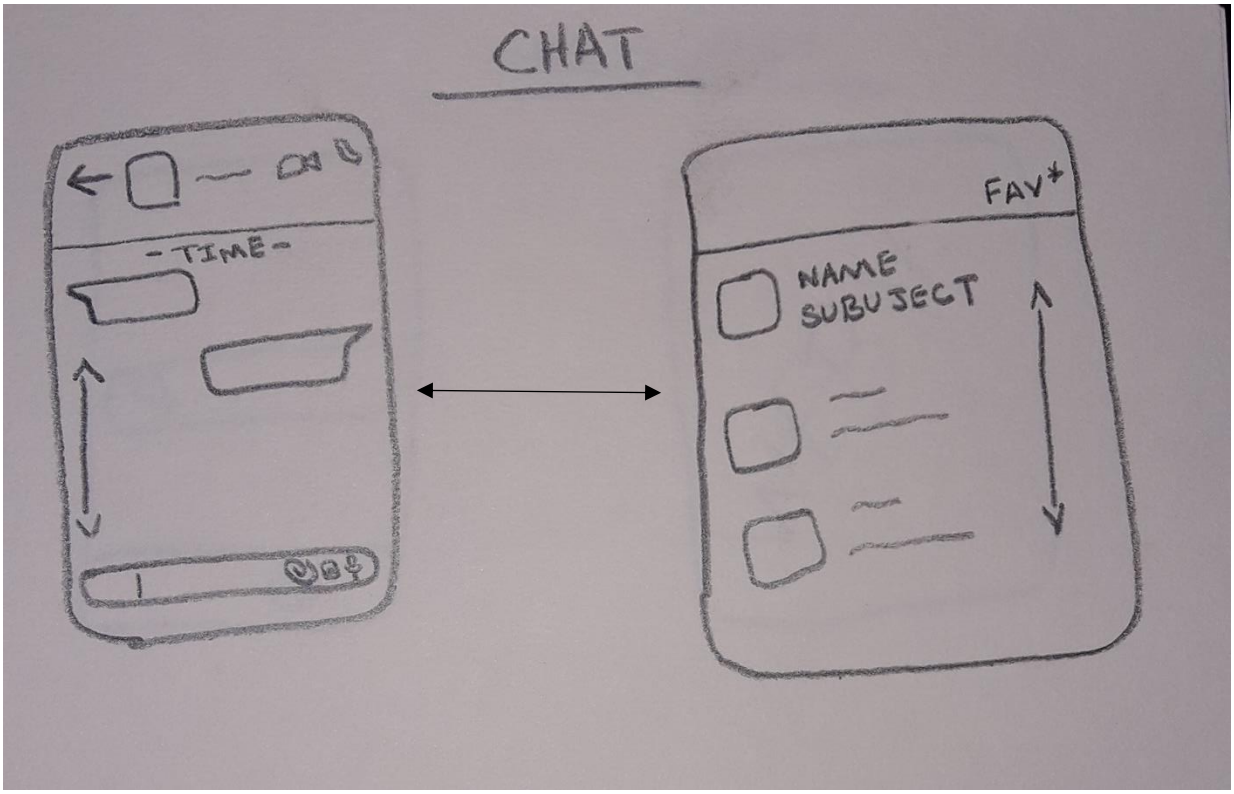
- A: add to favorites
- R: read now
- B: back
- D: download
- Hand drawn vertical arrow shows that the screen should be scrollable

- Hand drawn horizontal arrow represents a back button
- READ: in-app reader screen



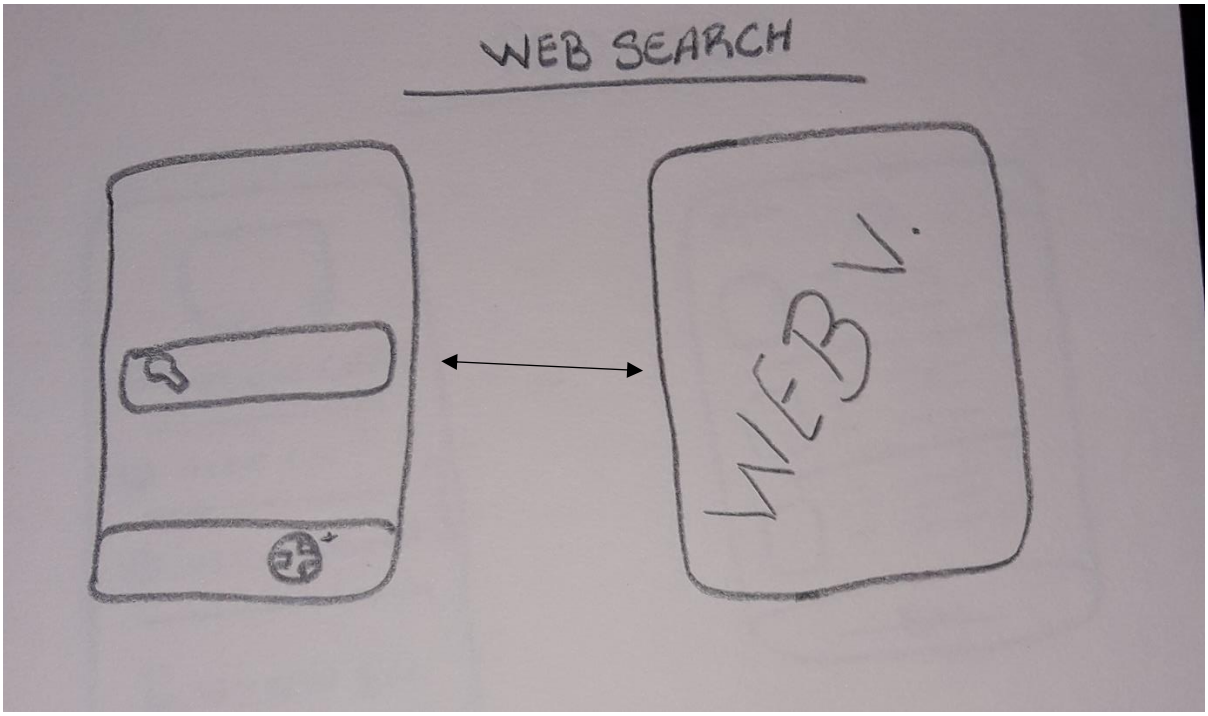
Key:

- A: accept
- C: cancel
- Hand drawn vertical arrow shows that the screen should be scrollable



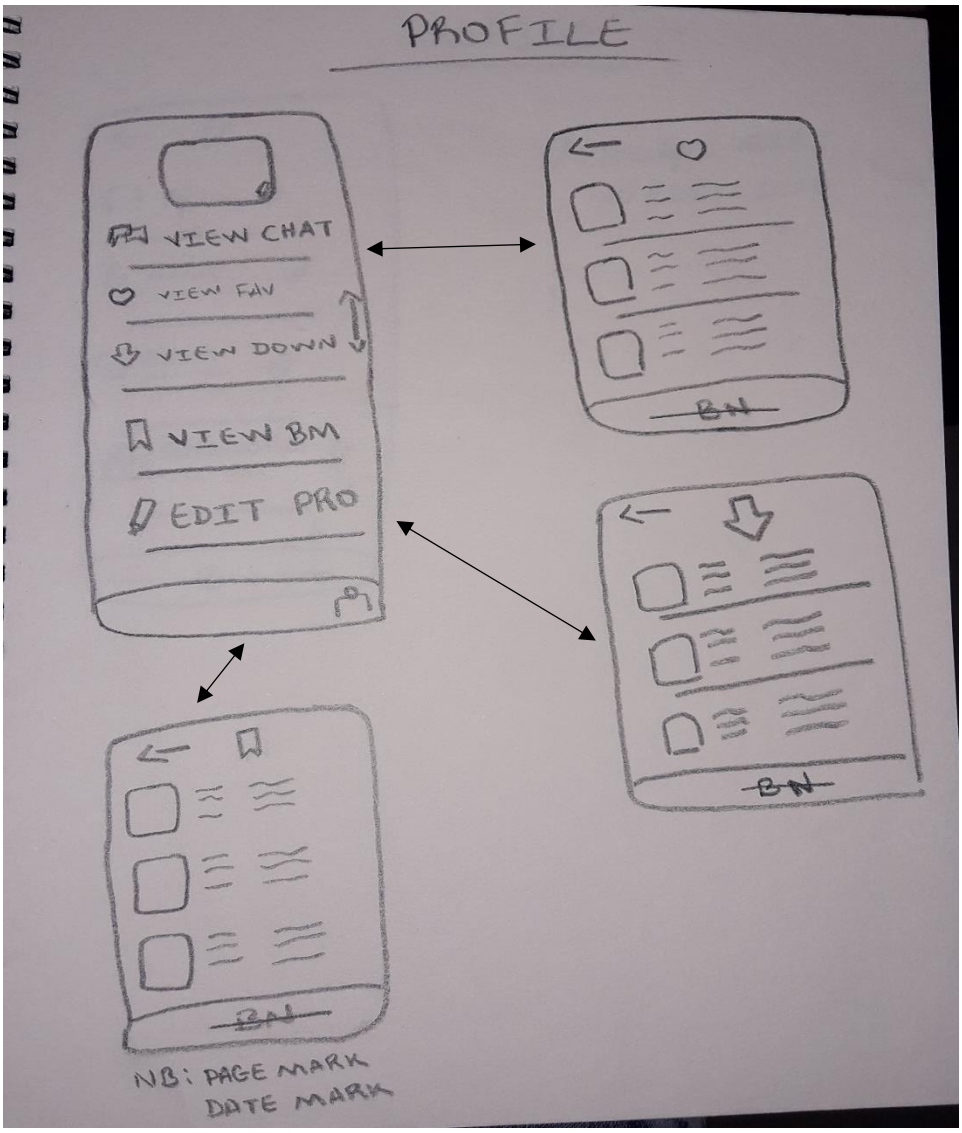
Key:

- Hand drawn vertical arrow shows that the screen should be scrollable
- Hand drawn horizontal arrow represents a back button



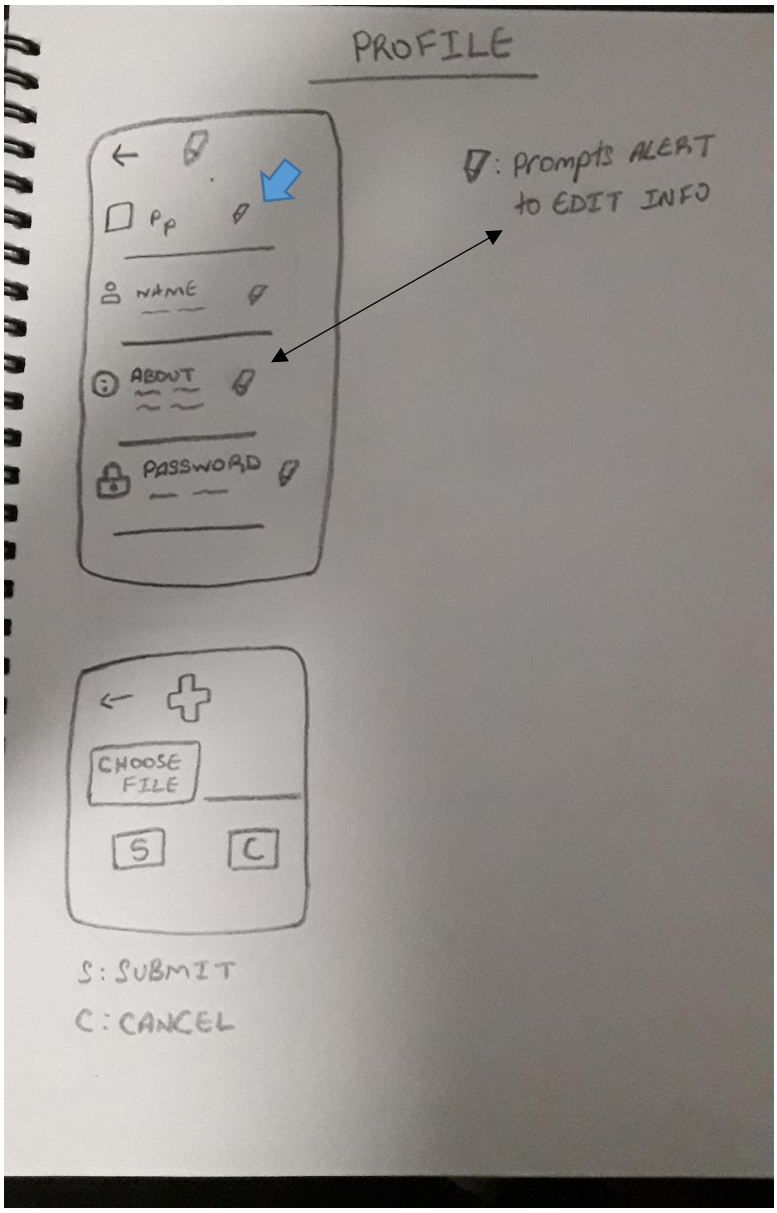
Key:

- WEB V: webview screen to show search results.



Key:

- Hand drawn vertical arrow shows that the screen should be scrollable
- Hand drawn horizontal arrow represents a back button



Key:

- Clicking on one of the pencils highlighted by the blue arrow will open an alert dialog box that will allow the user to edit the information they selected.
- Hand drawn horizontal arrow represents a back button
- PP: profile picture

- S: submit
- C: cancel

4.5.2 Screen Objects and Actions

As seen above the finished application will consist of roughly twenty screens. Each screen has specialized objects uniquely placed to allow the user to seamlessly traverse through the application. Listed below are all the known screens and the objects associated with them.

- login
    - Login button to submit credentials
    - Sign up button to move to sign-up page
- Sign up
    - Radio buttons to choose account type
    - Back button to go back to the previous page
    - Confirm button to submit information.
- Home screen
    - Bottom navigator with menu icons to traverse the application
    - Cards to allow for quick searches
    - Search bar to search the database for files

- Search results
    - List of search results
    - Read more button to move the reader to the read more screen
    - Favorite button to allow the user to add a particular file to their favorites
- Read more
    - Back button
    - Download button to allow the user to download the file
    - Read now button moves the reader to the in app reader
    - Favorite button
- Reader
- Chat home screen ***(Same layout as the home screen)***
- Chat search results
    - List of search results
    - Chat button to send chat request
- Chat screen(chat list)
    - Navigation bar with menu items to traverse through chat windows
    - List of previous chats
- Chat screen(chat requests)
    - List of chat requests
    - Confirm button to confirm request
    - Decline button to decline a request
    - Cancel button to cancel a request
    - Navigation bar with menu items to traverse through chat windows

- Chat screen(contact list)
    - Navigation bar with menu items to traverse through chat windows
    - Clickable list of liked tutors to open chat with a particular tutor
- Chat window
    - Text box to hold message before sending
    - Keyboard to type message
    - chat bubbles to hold message after sending
    - Tutor/student profile picture
    - Paper clip icon to send attachment
    - Mic icon to send voice notes
- Web search home screen
    - Bottom navigation same as the home screen
    - Search bar to perform web searches
- Web view
- Profile home screen
    - Bottom navigation same as the home screen
    - Clickable profile image that allows the user to view and change their profile image
    - List of clickable options to allow the user to move from window to window carrying out activities.
- favorites
    - back button
    - list of clickable items to bring the user to a particular file's read more page
- downloads

    - back button
    - list of clickable items to open a particular downloaded file
- bookmarks
    - Back button
    - List of clickable items to allow user to open a file at a particular location
- Edit profile
    - Back button
    - list of clickable items to allow the user to carry out activities edit their profile
- Upload file
    - Back button
    - Choose file button to allow the user to choose a file from their device to upload
    - Submit button to submit the selected file
    - Cancel button to cancel the action.

4.5.4 REQUIREMENTS MATRIX

Req. ID number	Name	Data structure	Components	description
REQ-1	Create Profile	Hash Table	Client Manager	The user should be able set up their profile either as student, tutor or publisher.
REQ-2	login	Hash Table	Client Manager	User should be able to login to the system using their sign-up credentials.
REQ-3	register	Hash Table	Client Manager	User should be able to create a new account if one doesn't already exist.
REQ-4	Search for document	Hash Table	Document Manager	The user should be able to search for scholarly documents and

				filter their search for multimedia files.
REQ-5	Read documents	n/a	Document Manager	The user should be able to read documents in app.
REQ-6	Play files	n/a	Search Manager, Document Manager	Users should be able to play multimedia files in the application
REQ-7	Filter search.	n/a	Search manager	The user should be able to filter their search.
REQ-8	Perform web searches.	n/a	Search Manager	User should be able to carry-out web searches from within the application.

				Users should be able to search for multimedia files
REQ-9	Send chat request	n/a	Chat Manager	Student users should be able to send chat requests to tutor and if need be cancel said request.
REQ-10	Save tutor	n/a	Client Manager	Students should be able to add preferred tutors to their favorites list
REQ-11	Send messages	n/a	Chat Manager	Users should be able to send text and multimedia messages.
REQ-12	Accept/decline chat request	n/a	Chat Manager	Tutor user should be able to

				accept or decline chat requests.
REQ-13	Upload files for peer review	n/a	Document Manager	Users should be able to upload documents for peer review
REQ-14	Upload files to DB	n/a	Document Manager	Users should be able to upload reviewed documents
REQ-15	Upload multimedia files	n/a	Document Manager	Users should be able to upload multimedia files
REQ-16	Peer review documents	n/a	Document Manager	Peer reviewer should be able to view documents that are uploaded for review

REQ-17	Send Notifications	n/a	Client Manager	The system should be able to send notifications to the user
REQ-18	Update profile	n/a	Document Manager	Users should be able to update their profile information.
REQ-19	Save to library	n/a	Document Manager	The user should be able to download documents/ multimedia files hence adding it to their library.
REQ-20	Make bookmarks	List	Client Manager	The user should be able to save bookmarks.
REQ-21	Find tutor	n/a	Search Manager	Students should be able to search for a tutor in a particular

				subject field in the chat.

4.6 DESIGN DECISIONS AND TRADE-OFFS

For the Edu.Share project, Docker was being considered for database deployment. This Idea was subsequently changed due to the fact that the application needs to be deployed and tested with persistent data in remote regions. To combat this this a choice had to be made to either purchase a local DBMS or utilize a free DBMS. Seeing as a local DBMS was not budgeted for the project, CleverCloud was chosen. CleverCloud however comes with its own issues. Using this DBMS the files cannot be uploaded from a local drive onto the service. Along with this CleverCloud only has 10 megabytes of storage and 5 connections. Due to the limited scope of this DBMS, it will only be using for testing purposes and replaced upon deployment of the application. For handling the chat communication between the Student and Tutor users, Socket.IO was chosen. This was chosen due to the relative ease of use to implement. This however has its own short comings. The initial connection when using Socket.IO is longer compared to WebSockets. Socket.IO is also not suitable for streaming that's data heavy by definition, for example video streaming.

The project also required the collaboration of a team of coders and for that it was thought best to use the platform provided by GitHub. The GitHub platform allows for users to collaborate on projects and aid in version control. No other platform was considered for this task due to the fact that the GitHub platform is one that is well known and the fact that it offers support to other

tools that the Edu.Share project utilises. Visual Studio code and Android Studio are two such tools. They both have Git embedded and allows for the Edu.Share project to have version control and collaboration through the GitHub platform. These two tools much like the GitHub platform was chosen due to the vast wealth of knowledge that was available about these tools.

MySQL Workbench was chosen as it works in collaboration with the CleverCloud tool. Through the use of this tool the Edu.Share project is provided with data modelling, SQL development and tools for server administration. Again due to the vast knowledge available on this tool no other tool was considered.

4.7 PSEUDOCODE FOR COMPONENTS

Review process:

file is submitted for peer review

invalidForm = **"**this form isn't valid**"**

noReviewers = **"**There is currently no one to review your file. Please try again later.**"**

termsDeclined = **"**you must accept our terms and conditions to submit file for review.**"**

If the submitter accepts the terms and conditions of the Edu.Share review process

subjectArea = file subject area

If users exist in the database who are registered as peer reviewers **and** their field of study matches the subject area variable

Reviews = 0

Score = []

Remarks = []

Add the file to the database

Add the file to those users' libraries to be reviewed

if a peer review form is submitted **and** valid

Reviews = **Reviews** + 1

Score.append(score**)**

Remarks.append(remarks**)**

Else

return invalidForm

Else

return noReviewers

If the number of reviews is equal to or greater than R **and** a period of time **T** is reached,

Score = sum(**Score**) / len(**Score**)

Print the variable contents **and** email to the file's submitter

If Score >= **S**

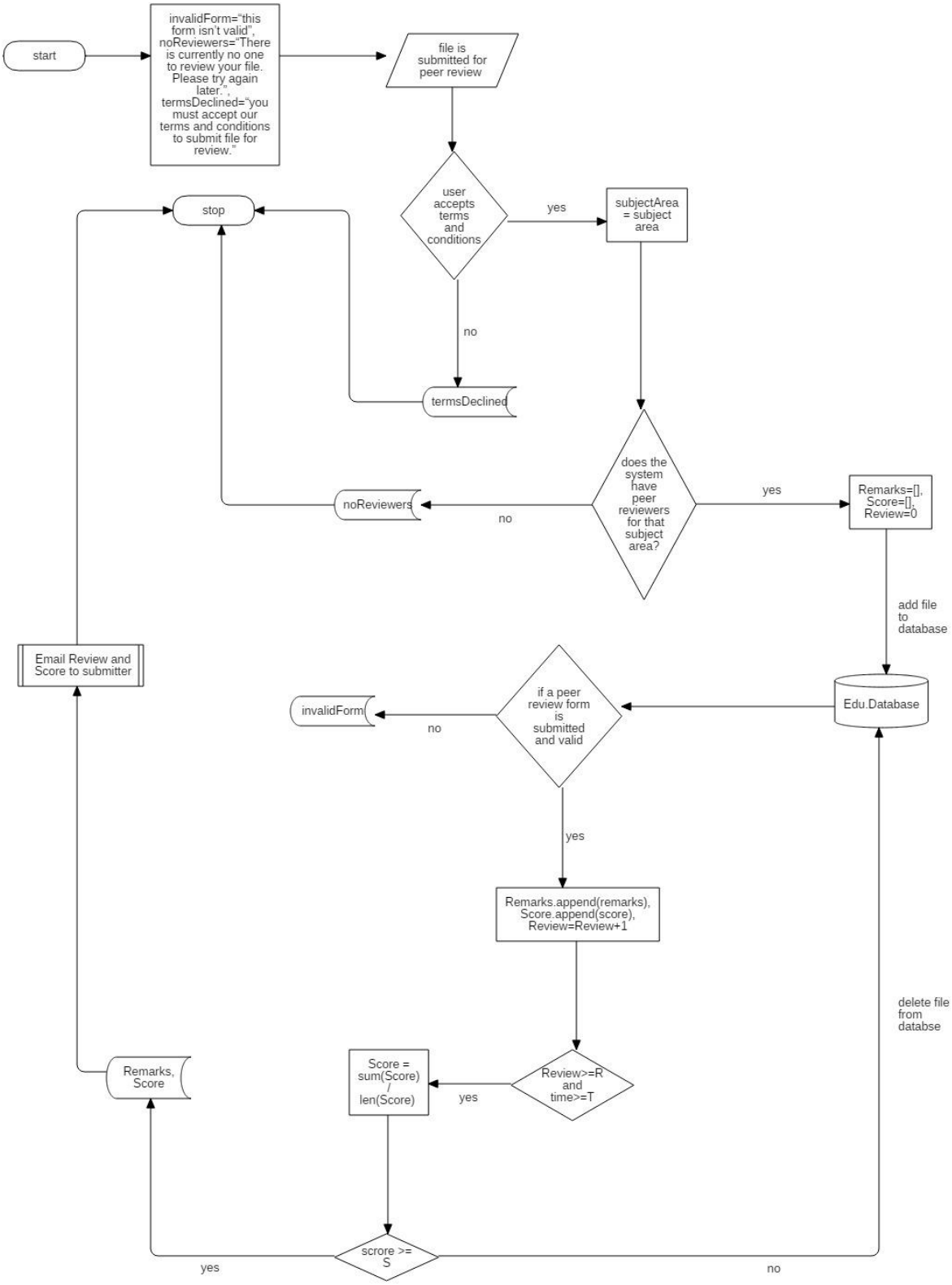
the file is added to the appropriate table.

Else

the file is deleted.

Else

Return termsDeclined


start
invalidForm="this form isn't valid", noReviewers="There is currently no one to review your file. Please try again later.", termsDeclined="you must accept our terms and conditions to submit file for review."
file is submitted for peer review
user accepts terms and conditions
yes
subjectArea = subject area
no
termsDeclined
stop
does the system have peer reviewers for that subject area?
yes
Remarks=[], Score=[], Review=0
noReviewers
no
add file to database
Edu.Database
if a peer review form is submitted and valid
invalidForm
no
yes
Remarks.append(remarks), Score.append(score), Review=Review+1
Review>=R and time>=T
Score = sum(Score) / len(Score)
yes
Remarks, Score
Email Review and Score to submitter
delete file from databse
scrore >= S
yes
no

Search:

user submits a keyword by using the search bar

Keyword = keyword

Results = []

For I in userInterestArea

For F in File table

If I matches the subject area of a file **and Keyword** can be found in the title **or** abstract of a file

Results.append(file**)**

Elif I and keyword do not match any file in the database

The search will be carried out with only the **keyword** variable

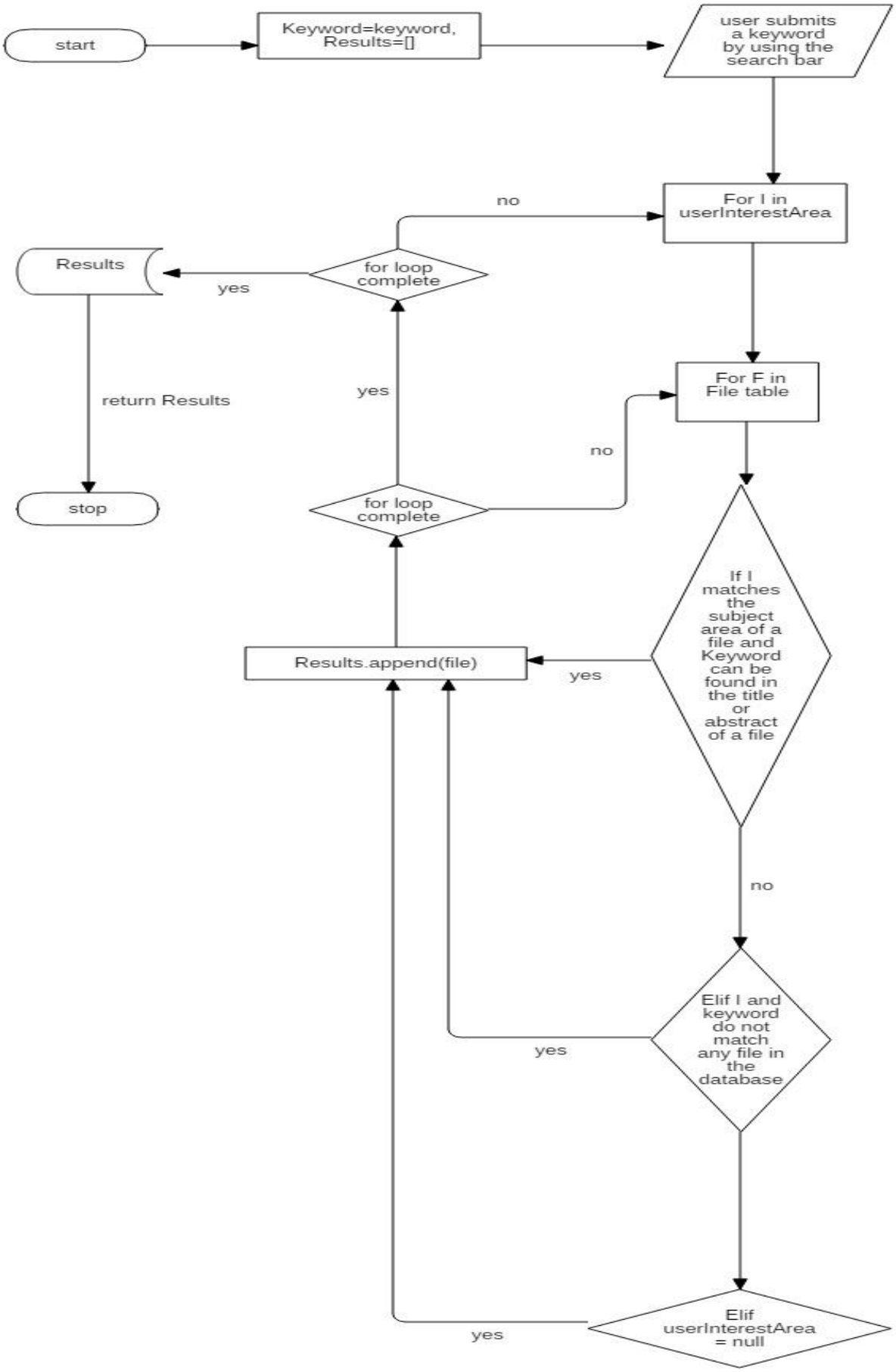
Results.append(file**)**

Elif userInterestArea = null

The search will be carried out with **keyword**

Results.append(file**)**

return Results


start
Keyword=keyword, Results=[]
user submits a keyword by using the search bar
For I in userInterestArea
no
for loop complete
yes
Results
return Results
stop
For F in File table
yes
no
for loop complete
If I matches the subject area of a file and Keyword can be found in the title or abstract of a file
Results.append(file)
yes
no
Elif I and keyword do not match any file in the database
yes
Elif userInterestArea = null
yes