Fall 12-9-2021

StudySnap

Liam A. Stickney
Sheridan College, stickney@sheridancollege.ca

Malik Sheharyaar Talhat
Sheridan College, talhat@sheridancollege.ca

Benjamin D. Sykes
Sheridan College, sykesb@sheridancollege.ca

STUDY SNAP

MOBILE COMPUTING | CAPSTONE PROJECT
HONOURS BACHELOR OF COMPUTER SCIENCE (MOBILE COMPUTING)

STUDENT TEAM

Benjamin Sykes, 4th Year Student
E: sykesb@sheridancollege.ca

Malik Sheharyaar Talhat, 4th Year Student
E: talhat@sheridancollege.ca

Liam Stickney, 4th Year Student
E: stickney@sheridancollege.ca
SUPERVISOR
Prof. Haya Elghalayini
E: haya.elghalayini@sheridancollege.ca
T: N/A
Sheridan College

DOMAIN EXPERTS

Jamie Goodfellow,
E: jamie.goodfellow@sheridancollege.ca
T: 905-845-9430
Manager of Learning Services,
Sheridan College

ABSTRACT

In today's modern education world, there have been several emerging technologies that were created to expand the learning experience beyond the traditional pencil and paper. Such items include online planners and numerous online student resource websites, but some students may find that there are sometimes *too* many resources. With so many resources available to them, students may find it overwhelming and may not even know where to start when it comes time to study. StudySnap aims to solve the unorganized and cluttered nature of study time by combining a clean resource organization tool with a smart and automatic resource search engine that provides students with resources created by their peers that are specifically tailored to their needs. Using various cloud computing services, natural language processing techniques, and Optical Character Recognition (OCR), students will be able to access a variety of notes about various topics. The system will analyze the user's interests or needs, and help find resources posted by other students that are specifically related to those needs. The user will then be able to select which notes they like, and view and rate the notes as needed. The goal is that by providing an all-in-one study system, students can remove some of the clutter that traditionally comes with studying and help get them the marks they are looking for.

ABOUT CAPSTONE PROJECTS

TIMELINE • PROGRAM • SCHOOL

- **January 2021 – April 2021**: Capstone Project Inception, 3-credit course (6 hours / week)
- **September 2021 – December 2021**: Capstone Project, 6-credit course (TBD hours / week)

PROGRAM • SCHOOL

- Hons. Bachelor of Appl. Computer Science (Mobile Computing)
- Applied Computing, Faculty of Applied Science and Technology

Sheridan | Get Creative

Table of Contents

INTRODUCTION

This document presents a brief overview of several key areas of the project, including some preliminary research about the project and its impact, as well as the various technologies and architecture that were used to develop StudySnap and its main use-cases. The following sections give an overview of the project and its impact, the project requirements, project architecture, project planning (including project and risk management), and the testing and validation strategies used. The following document changelog lists relevant dates and updates to this document as it evolves throughout the project's development.

Document Changelog

- **January 21st, 2021 –** Began work on the PID. Completed most of the preliminary work, which includes the introduction and overview (domain/industry, problem/solution descriptions), relevant technologies to the solution, the impact of the solution and the feasibility of the solution. The document's abstract was also completed at this time.
- **February 11th, 2021 –** Edit document with fix suggestions provided in initial feedback from project supervisor.
- **February 27th, 2021 -** Made small changes to update the iteration logs and updated other additional areas as the development of the project continued.
- **March 18th, 2021 –** Added the project requirements section, which included adding screenshots of the wireframes for our relevant use-cases. Additional screenshots of SwiftUI will be added in a future change.
- **March 19th, 2021 –** Made small changes to the project overview section to reflect the developments made during the architectural release.
- **March 20th, 2021 –** Added to the project requirements section with descriptions of the main use-cases to be implemented as well as screenshots of the use-case's wireframes and SwiftUI implementations.
- **March 25th, 2021 –** Added to the project architecture section an overview of the project's architecture, the main components that make up the architecture as well as a description of the deployment model. Screenshots were also added for clarity.
- **April 2nd, 2021 -** Went through each section and updated information as recommended by the supervisor in a live meeting.
- **April 9th, 2021 –** Added the project validation section and began work on the conclusion section.
- **April 11th, 2021 –** Finished the conclusion section of the document.
- **April 13th, 2021 –** Added technological references to bibliography.
- **April 16th, 2021 –** Fixed formatting and finalized PID submission.
- **September 9th, 2021 –** Updated PID sections according to the received defense feedback
- **September 16th, 2021 –** Updated Project Plan section to include a brief summary of changes made to the Jira project plan and revised previous sections based on feedback from the first assignment
- **October 11th, 2021 –** Updated the document to reflect work done during the iterations inside the alpha release
- **October 13th, 2021 –** Updated the document's Use-Cases and User Interface sections to be up to date with the current UI and implemented use-cases. Removed references to the tag-bases search as the use-cases related to it have changed. The tag-based search was more useful when the application was being designed a as a general knowledge application. With the addition of the classroom feature, the application is more

focused on providing specific references to what the user searches for, rather than a broader recommendation. As such, the tag-based search and automated recommendation engine have been removed.

- **November 14th, 2021 –** Updated the use-cases completed to include the new note rating feature, and updated some of the design model diagrams to reflect changes to the cloud architecture
- **November 24th, 2021 –** Updated the use-cases completed to include the citation features, and went through many of the areas of the document to remove features that were considered out of scope and update the project's architecture. Also updated the use-case images to reflect the newer UI
- **December 2nd, 2021 –** Updated several areas of the document to use proper tense and removed some references to cancelled features
- **December 4th, 2021 –** Added a new use-case as it was implemented on the front-end: edit note
- **December 7th, 2021 –** Updated the user testimonials section as well as the validation section

PROJECT OVERVIEW

StudySnap aims to target the problematic nature of having too much information to study, but not enough information on where to start. With the number of learning resources currently available to students, whether they be websites, textbooks, academic journals, etc., students can often become overwhelmed at the amount of information available to them. They may have to spend large amounts of time reading through material, only to find out that the information they've just read isn't helpful to them. This not only wastes time but can cause unnecessary stress and anxiety. The project solves this issue by including not only an easy-to-use, collaborative **note organization tool**, but also an **automated and in-depth note search tool,** so students can spend less time searching for resources they need and more time actually studying them. The team members include Liam Stickney, Benjamin Sykes and Malik Sheharyaar Talhat, all of which have been very keen on trying to leverage natural language processing techniques, Optical Character Recognition (OCR), and cloud computing services into this solution. The following subsections introduce information related to the Educational Services industry, description of the problem and the proposed solution, technologies that will be used to construct the solution, the expected impact the solution is expected to have and a brief analysis on how feasible the solution is to incorporate into the domain.

DOMAIN AND INDUSTRY OVERVIEW

As the project supplies educational aid to students not directly related to the academic institutions themselves, this project falls under the Educational Services industry, but more specifically, Educational Services (61) \ Educational Services (611) \ Educational Support Services (6117) \ Educational Support Services (61171) [1]. The Canadian Industry Statistics website defines this industry as "this industry comprises establishments primarily engaged in providing non-instructional services that support educational processes or systems" [1]. As of 2019, there were 5,785 businesses in Canada under this industry, with 98.2% of them containing 99 or less employees [1]. In fact, 53.9% of these businesses were considered "micro" meaning that they had 5 or less employees [1]. There are only 2 businesses in all of Canada under this industry that are considered "large", meaning they have 500 or more employees [1]. This would suggest most businesses and establishments in this field are very small groups of people looking to support students' educational performances, not unlike this project. In terms of financial performance, among all businesses in this industry (the bottom quartile, the low middle, the high middle and the top quartile), the average total revenue is 179.5 thousand dollars, which Statistics Canada rates as "very good" [1]. The average total expenses are 130.2 thousand dollars, which is also rated as "very good" [1]. This gives an average net profit of 49.3 thousand dollars, which is also rated as "very good" [1].

In terms of existing applications that are already present in the domain, there are several popular study applications present In the Educational Services industry. One such example is a mobile application called Chegg Prep, which is an app that allows students to view a large number of "flashcards" that are designed to test the student's knowledge on a variety of subjects. This application also utilizes the idea of students creating and uploading their own content for other students. StudyBlue was also an application that was very similar to Chegg Prep, until it was bought by Chegg in late 2020. StudyBlue functioned very similarly to Chegg Prep, in that students could create their own study notes and quizzes and upload them for other students to view. As of October 26th, 2020, StudyBlue was the 9th ranked educational application on the App Store, with over 50 million active monthly users. It had an average rating of 4.7 out of 5 stars with over 17,964 ratings. Clearly, the domain contains similar solutions to the one we are proposing, and as such, there is a clear demand for the features present in our solution.

PROBLEM DESCRIPTION

The problem that this project solves is the traditionally cluttered nature of classes from the perspective of the student. As a student grows older and older, the amount of work given to them usually increases, while the amount of "hand-holding" provided decreases. For some, this is not a problem, but for many students, the transition from secondary to post-secondary or even elementary to secondary can be a stressful ordeal. These transitions are typically when students are expected to know how to operate individually and measure what is needed to obtain their own success. In doing so, students can sometimes lose track of what is expected of them and quickly fall behind in their studies. With the advent of many emerging technologies, this problem is further developed, as students can quickly become bogged down with *too* many study materials, such as lecture notes or research articles and not enough time. While students may have a lot of resources available to them, many of them are usually not related to what the student is looking for at that time. The process of **frantically searching through resources** can be anxiety-inducing, as the student wastes more and more of their time looking for what is helpful to them instead of getting their work done. Constantly being in a state of disarray and stress induced by these conditions can certainly have a negative impact on one's mental health. A study conducted in 2018 concluded that self-perceived academic performance was the main cause of suicide among students in Mexico, and that the rate of suicide among those aged 15-29 has been increasing rapidly [2]. Clearly there is evidence that suggests that there is a direct correlation between student's mental health and their performance in school. While it is unrealistic to expect that a study tool such as StudySnap would be able to fully solve these complex issues, we hope that it can help students by allowing them to organize their time more efficiently and in doing so, help lower the stress of studying while simultaneously increasing their academic performance. Such an impact would hopefully help the student body, and in doing so, increase the academic performance of several academic institutions.

SOLUTION DESCRIPTION

The solution solves the problem described above by providing a system that not only allows students to **organize their notes** in a single place, but also **supplies students with resources that are tailored to their own personal needs**. There already exist many applications that allow students to view notes created by other students. The problem with these pre-existing solutions is that none of them solve the problem stated in our problem description. Now, a student may need to look through many articles or files before they come across something that is even remotely related to what they were originally looking for. StudySnap's **search engine, developed using natural language processing techniques**, fixes this problem by analyzing many documents and individually picking out information that the user is looking for. The resources supplied to the user all come from other users who have uploaded their own unique notes to the cloud for fellow classmates to access. For the system to pick the most relevant documents, a **rating scale** has been put in place that allows users to rate uploaded notes based on their accuracy and helpfulness. Furthermore, the notes are logically sorted into **classrooms,** which users can create to encourage collaboration between their classmates and ensure the notes they view are directly related to what they've learned in class. This promotes a mutually beneficial learning environment in which students can achieve a higher level of study while simultaneously improving the studies of their classmates. Finally, StudySnap is deployed as an iOS mobile application. As many (if not all) students now own a smartphone, being deployed as a mobile application will target a larger audience and allow the users to always keep the application on them. The following subsections detail the technologies that will be implemented to produce the solution.

Mobile Computing

The Study Snap mobile app is the key component in providing students with a convenient and always available solution to their on-the-go study routines. Obviously, it goes without saying that most of the population, especially people between the ages of 15-30, have some sort of smart device, whether that be a tablet or a phone. By offering the application on a widely used set of devices, it can be ensured that the largest possible audience is targeted. Furthermore, phones and tablets have important features built into them like a file storage application and a camera, which can be used within the project for additional features.

Cloud Computing

Cloud computing provides an important supporting role as it is required to allow the users to upload their notes to share with others. Users can then use the search engine to view other user's notes related to their search and can then rate each other's notes based on their accuracy and helpfulness. StudySnap's primary concern is to provide fast, effective study material discovery to its users. With cloud, the application leverages the numerous cloud-powered cognitive computing services, as well as storage and compute resources. Cloud is essential for implementing several key features of the application, in addition to hosting the application deployments.

Advanced Areas of Computer Science

An incredibly accurate search system is a must in helping collect and analyze data from notes, either digital or handwritten, that are uploaded in the process of creating a note. The analysis and extraction of this data, done through the use of **cognitive computing-based natural language processing techniques,** ensures the most relevant notes are provided to the user. Using this extracted metadata, **artificial intelligence-based elastic search methods**, available from the cloud computing resources, it provides the users with relevant study material fully customized to their current needs. Furthermore, the application makes use of **optical character recognition (OCR) techniques** to analyze hand-written notes and convert them to a digital format.

SOLUTION IMPACT

The proposed system is intended to provide a more defined solution to several existing problems that are being only partially addressed by existing applications on the market. As discussed briefly, the applications already present in the domain do contain similar features to this solution but fail to solve the problem we are targeting, that being the lack of a method to obtain the most relevant study material automatically while encouraging a collaboration-based level of learning. A main problem that students face while studying is that they do not know which materials available to them are the most relevant to what they want to study. For instance, an application like Chegg Prep supplies the students with additional study material written by other students but does not provide an easy way to distribute that content to the student besides a manual search. To illustrate this, consider the way a professor would assign reading material to a group of students. The professor has the exact context of what the students need to study, since he/she is the one creating the tests or exams. An application like Chegg Prep simply gives the student a large number of resources, and essentially asks the student to figure the rest out for himself. The solution here enhances the searching process by utilizing an **artificially intelligent text-based search engine** such that the student no longer has to spend time figuring out what to study himself. Furthermore, the **logical classroom groupings** ensure that the notes the user searches for are directly related to what they've learned in their own classes, all while encouraging collaboration between classmates. As for the actual benefits the proposed solution provides, there can be an increase in both mental wellbeing and academic performance among students who benefit from the automatically recommended notes.

SOLUTION FEASIBILITY

This subsection discusses some of the feasibility considerations associated with the project and its construction.

Design and Construction

The focus of this project is very heavily weighted on its ability to provide accurate and relevant search results when using the **built-in text-based search engine**. The system stores student account data and note data through its PostgreSQL database and through cloud storage solutions for files that are uploaded along with student/user notes.

With the accessibility of cognitive, cloud storage and cloud compute resources available through the cloud providers, the overall design and construction challenges are very feasible, with the only risks being costs for such resources getting out of hand. Most technological risks associated with deployment and cloud resources, according to the proposed system architecture, were mitigated easily by handling migrations to other more suitable tools as limitations were determined.

Deployment

By leveraging the GitHub Actions CI/CD platform, seamlessly integrating code changes with tests (unit and end-to-end) in addition to deployment processes came at very little cost to sprint deadlines. Furthermore, by utilizing containerized solutions packaged with helm, deploying code with bug fixes and features to the Kubernetes infrastructure was fast, automated, and results were obtained in virtually zero downtime to the end-user. This is done by using rolling deployments, which takes advantage of a multi-node cluster by creating a copy of the previous running instance of the application while the new version is being deployed.

Adoption

Students are fast adopters of modern technologies and are quick to adapt to anything that can improve the efficiency and quality of their schoolwork. For this reason, it is believed that with a low/no-cost price model and the application being available on major application distribution platforms, students will quickly adopt and take advantage of StudySnap's many helpful features which work to make managing schoolwork easier.

PROJECT REQUIREMENTS

This section introduces details of some of the main project requirements in a broad overview. For more details check out the Visual Paradigm diagrams and models available through vPository.

SYSTEM CONTEXT

As of the current implementation, StudySnap is planned to only have a single primary end user, that being students enrolled in secondary or post-secondary educational institutions. As such, there is only a single primary actor, that being the student(s) themselves. These users will be the ones who directly sign-up to StudySnap, and will also be the ones who upload, organize, view and search for notes, as well as create and join classrooms.

The main summary use-case is defined as "publish, organize and search for student study material". This definition gives an extremely high-level overview of what StudySnap's main goal is, while also alluding to three smaller functions of the system, which include organizing and sorting student study materials, uploading and viewing study material, and searching for additional study materials. With these three pieces in mind, the system is broken into three major functional areas, each one containing a summary use-case represented by each of the three pieces of functionality just mentioned.

Functional area one contains the pieces of the application that allow students to sort and organize their study material, which is referenced by this area's functional use-case: "manage notes".
Functional area two contains the requirements of the application that handle the ability to upload new notes to the application, as well as view previously uploaded notes, either by the same user or uploaded by other users. The summary use-case of this functional area is given by: "view and publish notes".

Functional area three contains the requirements of the application that allow users to search for additional study material, which mainly concerns the text-based note searching capabilities. The summary use-case for this area is "find and review study material".

These three functional areas are defined in the system overview diagram. As for the other actors involved in the system, there are two supporting actors and two offstage actors that have been identified. The two supporting actors are a teacher/professor and a storage system. The storage system is essential to the system as it allows user accounts and their uploaded content to be stored in a location where it can easily be accessed. A teacher or professor may create an account to upload content for students to view, review student uploaded quizzes, create their own quizzes, or may be contacted if their work is to be cited in the application, however, there is no current implementation in place that allows teachers to share study material directly with students. Finally, for the two offstage actors, there is an academic institution and a third-party learning center. These are actors who would likely never actually interact with the system, however, may be influenced by its existence if some sort of licensing deal was made between StudySnap and another company.

USE-CASES

In order to demonstrate the major areas of the system, all of the use-cases currently implemented are listed below along with brief descriptions of each one.

Use-Case	Functional Area	Description
Create Classroom	1	The user can create a new classroom, a separate area in which users can share private notes with other users in the same classroom. Creating the classroom will generate a private code, in which other users can enter to join the same classroom. The classroom and its ID are stored in a separate database table.
Delete Classroom	1	The owner of a classroom should be able to delete the entire classroom from within the specific classroom's view. The system should also then delete all the notes stored within the classroom and remove all other users from the classroom.
Delete Note	1	A user should be able to delete a note that they previously uploaded. Deleting the note should remove it from both the classroom it was uploaded in as well as from the user's personal note view.
Join Classroom	1	The student can use a generated code provided by another student to join a classroom. From within the classroom, users can view other notes uploaded by users in the same classroom, as well as discuss other class-related ideas.
Leave Classroom	1	If the user wants to leave a classroom, they should be able to enter the classroom and click a leave button,

		which will allow them to leave. Doing so will remove them from the classroom.
Edit Note	1	The user should be able to view a note they have previously uploaded in their personal storage view, and change both the name of the note and the description. The note's name/description should then appear changed across the whole application.
Publish Note to Cloud	2	The user wants to upload a note (whether it be using a digital file or physical image of a note), annotate it with some information (keywords, title, description, etc.) and upload it to the cloud in a specific classroom that they are inside of.
Scan Physical Note	2	A user wants to scan a picture of a physical note during the upload process. Doing so will utilize the camera, and use OCR to extract the text from the image. From there, the user can edit the scanned text to fix any errors, and then convert it into the applicable file type. The note is then uploaded as normal.
Select Digital Note	2	During the upload process, the user should be able to select a digital note present on their device. This involves accessing the device's files and allowing the user to search for the file they wish to upload. Once a file is selected, the upload process should continue as normal.
View a Note	2	The user wants to click on an uploaded note and view some information about it. This includes things like title, author, description, body, etc. The user should be able to view both his/her own notes and notes uploaded by other users, as long as the user is within the same classroom.
Rate a note on the cloud	2	While viewing other user's notes in the same classroom, the user wants to add his/her rating to the note based on its helpfulness and quality. The user needs to enter his/her rating (out of 5) and submit the rating to update the rating on the cloud.
Create/view citation	2	During the note upload, the user should be able to enter in some citation information to cite the original author they have taken information from. Conversely, when viewing a note, other users should be able to view citations that were included when the note was uploaded.
Search for Notes Using Text-Based Search	3	From within a classroom, the user should be able to enter a query to search for uploaded notes. The system should utilize elastic search to find

		semantically similar notes in that classroom, and return a list of notes in descending order of relation to the query.

USER INTERFACE

In this section, the user interfaces for some of the above use-cases are shown and described. The use-case in relation to the user interface is also described.

For the delete note use-case, the user must simply navigate to the note organization page. For this use-case, the user needs to be able to delete a note as it appears in the organization system. The user simply needs to swipe left on the note he/she wishes to delete. This will display a delete button next to the note. The user then needs to simply click on the delete button and the corresponding note will be deleted in the note database. See Figure 1: UI for deleting notes for images of the note storage and the note delete process.

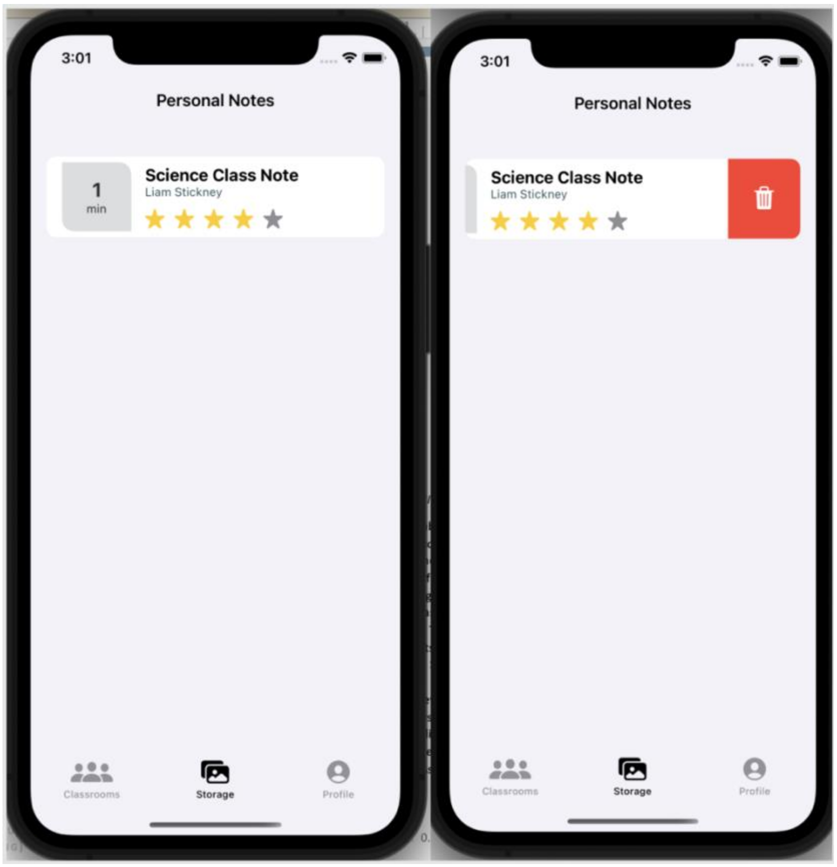
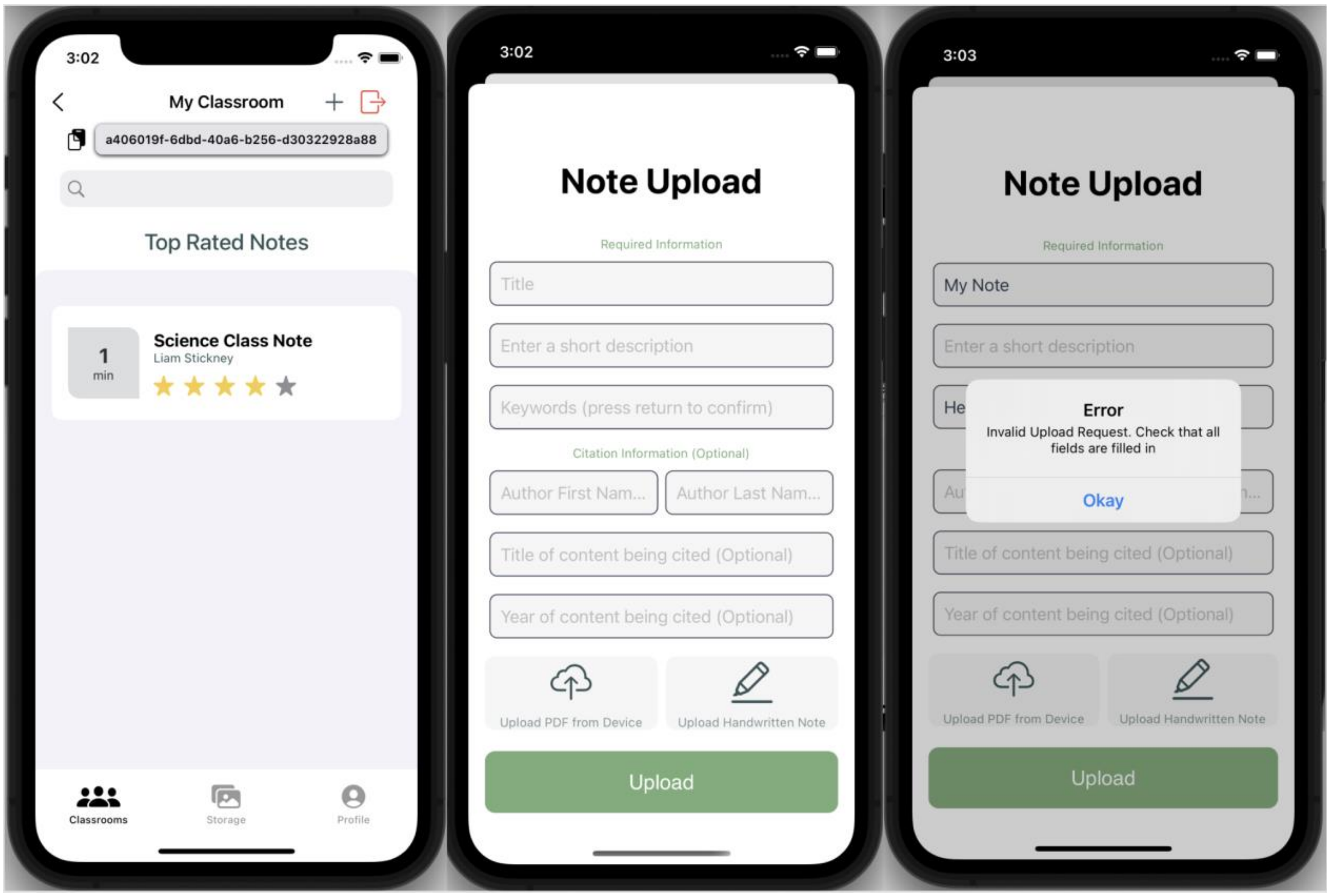


Figure 1: UI for deleting notes

For the publishing a note to the cloud use-case, the user must first navigate to a classroom their enrolled in and click the plus icon. From there, the user will be taken to a screen in which they can select the kind of note they wish to upload, and enter in any other required information, like title, keywords, description, etc. They can also provide citation information if they wish. At some point during the upload process, the user will need to upload either a preexisting PDF file, or take a picture of a physical note. Once the user has entered all the required information, and a PDF file has been uploaded, the system will confirm that all the inputs are valid and proceed to upload the note to the cloud. The classroom the user is uploading a note to represents a single, real-life course the student is enrolled in, so students are advised to upload notes into classrooms that share similar content to the contents of the note being uploaded. See Figure 2: UI for note upload for images of the note uploading process.

For the viewing of a note use-case, the user can either choose to click on a note from their own personal note storage system, or click on a note that appears after the result of a note search in a classroom. In either case, once the user clicks on a note, the system will find the corresponding note in the cloud, and return the information found back to the application to display to the user. There will also be an option to rate the note, as well as view the full PDF file associated with the note. See Figure 3: UI for note viewing for more details.


3:02
My Classroom
a406019f-6dbd-40a6-b256-d30322928a88
Top Rated Notes
1
min
Science Class Note
Liam Stickney
Classrooms
Storage
Profile
3:02
Note Upload
Required Information
Title
Enter a short description
Keywords (press return to confirm)
Citation Information (Optional)
Author First Nam...
Author Last Nam...
Title of content being cited (Optional)
Year of content being cited (Optional)
Upload PDF from Device
Upload Handwritten Note
Upload
3:03
Note Upload
Required Information
My Note
Enter a short description
Error
Invalid Upload Request. Check that all fields are filled in
Okay
Title of content being cited (Optional)
Year of content being cited (Optional)
Upload PDF from Device
Upload Handwritten Note
Upload
Figure 2: UI for note upload

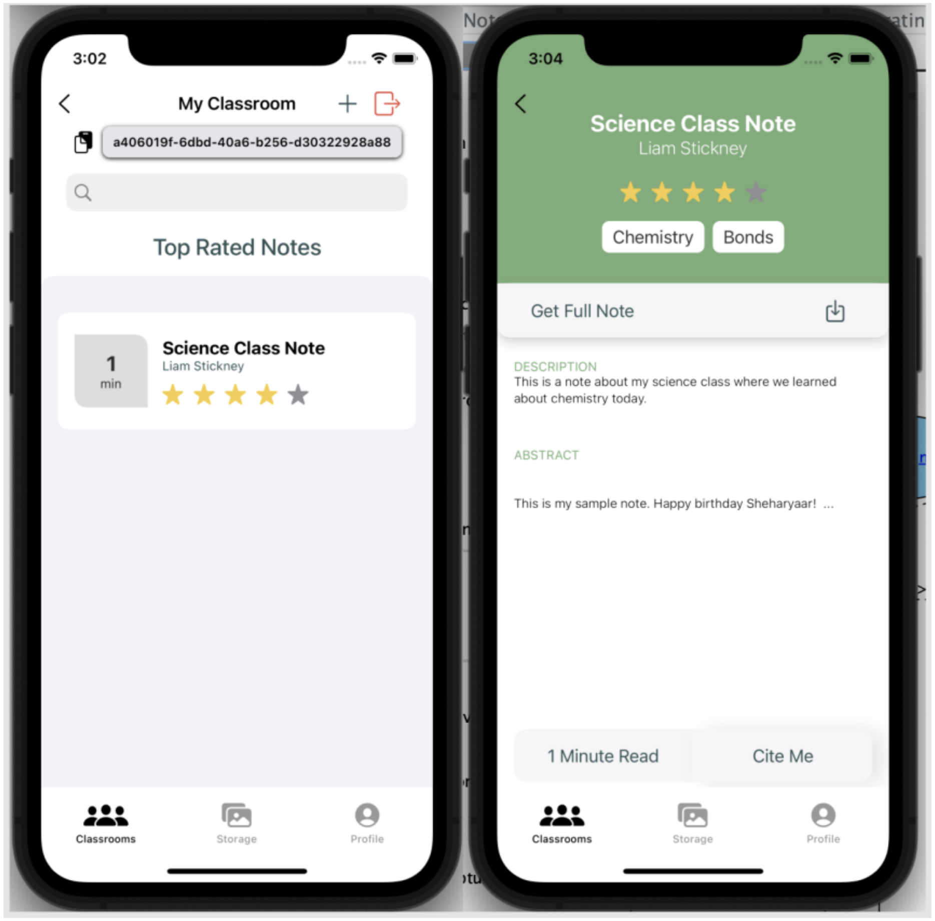


Figure 3: UI for note viewing

Finally, for using the text-based search, the user just needs to navigate to a classroom to access the text-based search. From there, it is as simple as entering some query that the user wishes to search for, and the system will utilize a text-based search to search through a set of notes in the StudySnap database based on the entered query. Once the system has identified a set of notes that matches the query, a list of notes is returned back to the application, at which point they are listed in a list view that the user can click on to view the note in more detail (as described in the use-case above this one). See Figure 4: UI for text-based search for images related to the text-based search process.

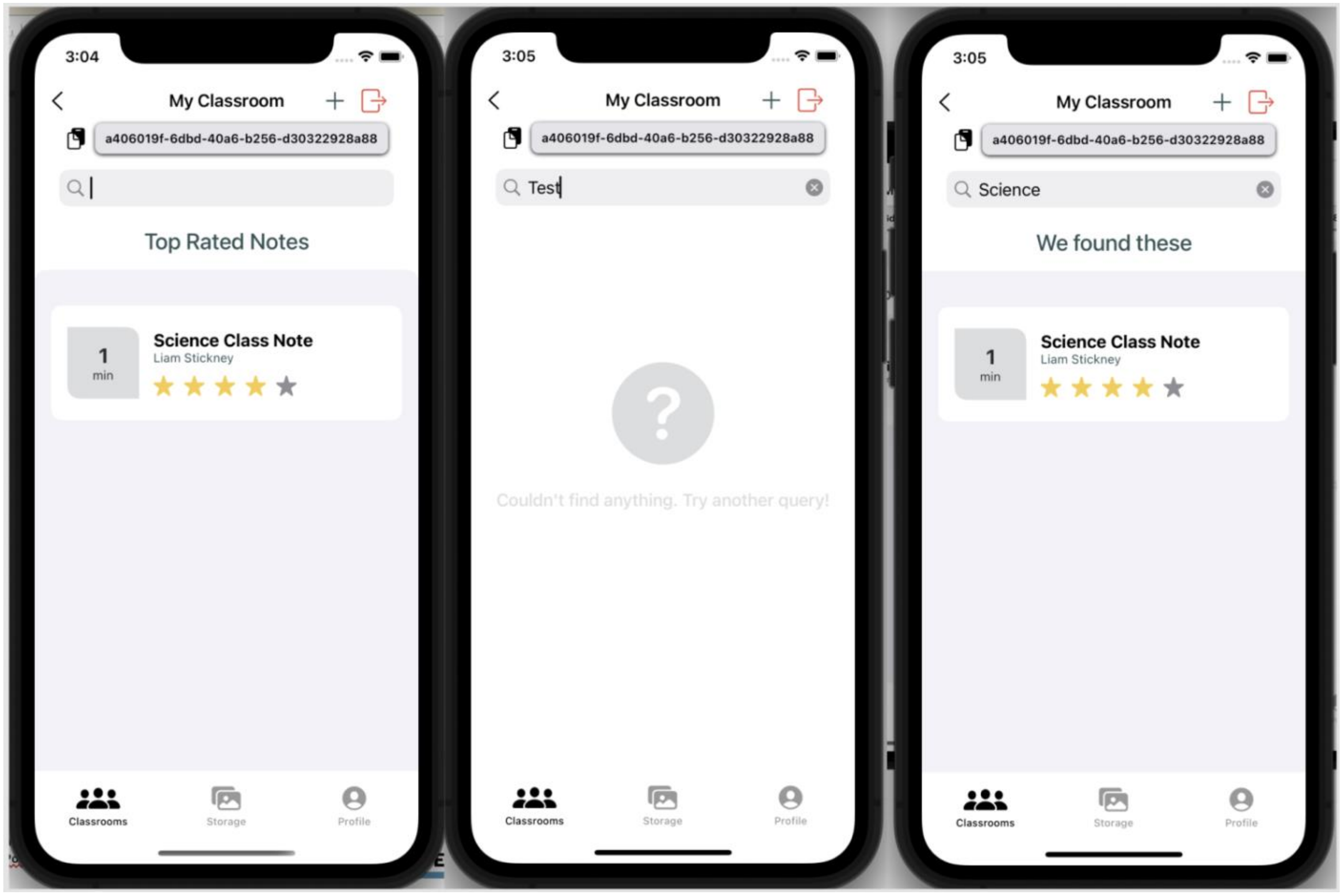


Figure 4: UI for text-based search

PROJECT ARCHITECTURE

This section introduces concepts, design considerations, and the overall system representation for StudySnap's software system architecture. The software model contains the design model, interaction model, and deployment model, which can be found in the Visual Paradigm online repository (VPository) located at the following address: Click Here.

ARCHITECTURE OVERVIEW

The overall architecture of StudySnap is based on a microservices architecture, as depicted in Figure 5: Architectural overview diagram below, which consists of 3 key components that make up the system. The system starts when the client makes a request from the StudySnap client application to the **API gateway**, also known as the **Load Balancer**. The Load Balancer acts as the primary ingress traffic handler which allows the system to expose certain applications, like the public-facing API endpoints, or other microservices to the client application(s). Once the request is propagated to the Load Balancer, the Load Balancer can then distribute the ingress traffic to the target services from inside the cluster, which includes the **authentication/authorization service, Neptune and Logstash**. Neptune, the primary resource service, has access to the Elasticsearch service and the **note table** (contained within the StudySnap database), which handles much of the exposed full-text and semantic search capabilities of the system. Finally, the microservices cluster provides protected access to the important and sometimes secure/sensitive data sources, which includes the **Elastic Search Index,** and the main **StudySnap database**. By hiding the private services behind the cluster, the system avoids exposing sensitive data and underlying implementation details to potential threats outside the cluster and only allows access to services that require direct public-facing connections.

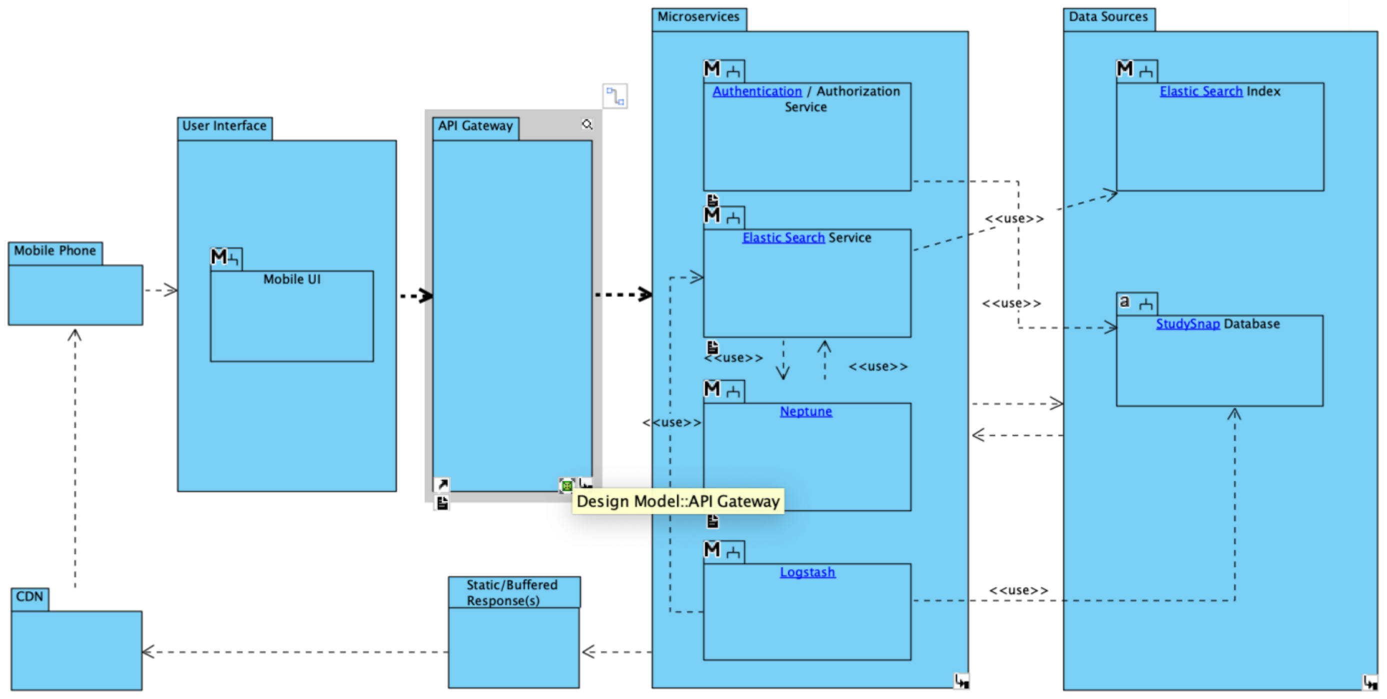


Figure 5: Architectural overview diagram

SYSTEM COMPONENTS

As discussed in the architecture overview, the overall system architecture consists of many small micro services which make up the functionality of the application. This functionality can be broken up into the API Gateway (Load Balancer), microservices, as well as any data sources and storage accounts that help provide and support business logic in the system. Normally, the client application would also be part of this architecture, however, since the client application is an iOS application and cannot be hosted within the cluster, the relationship between the client iOS app and the services acts as a hybrid between microservices architecture and traditional client-server architecture. As seen in Figure 6: Architecture diagram of authentication/authorization microservice, each of these larger components can be broken down into the smaller services and classes that make up that component.

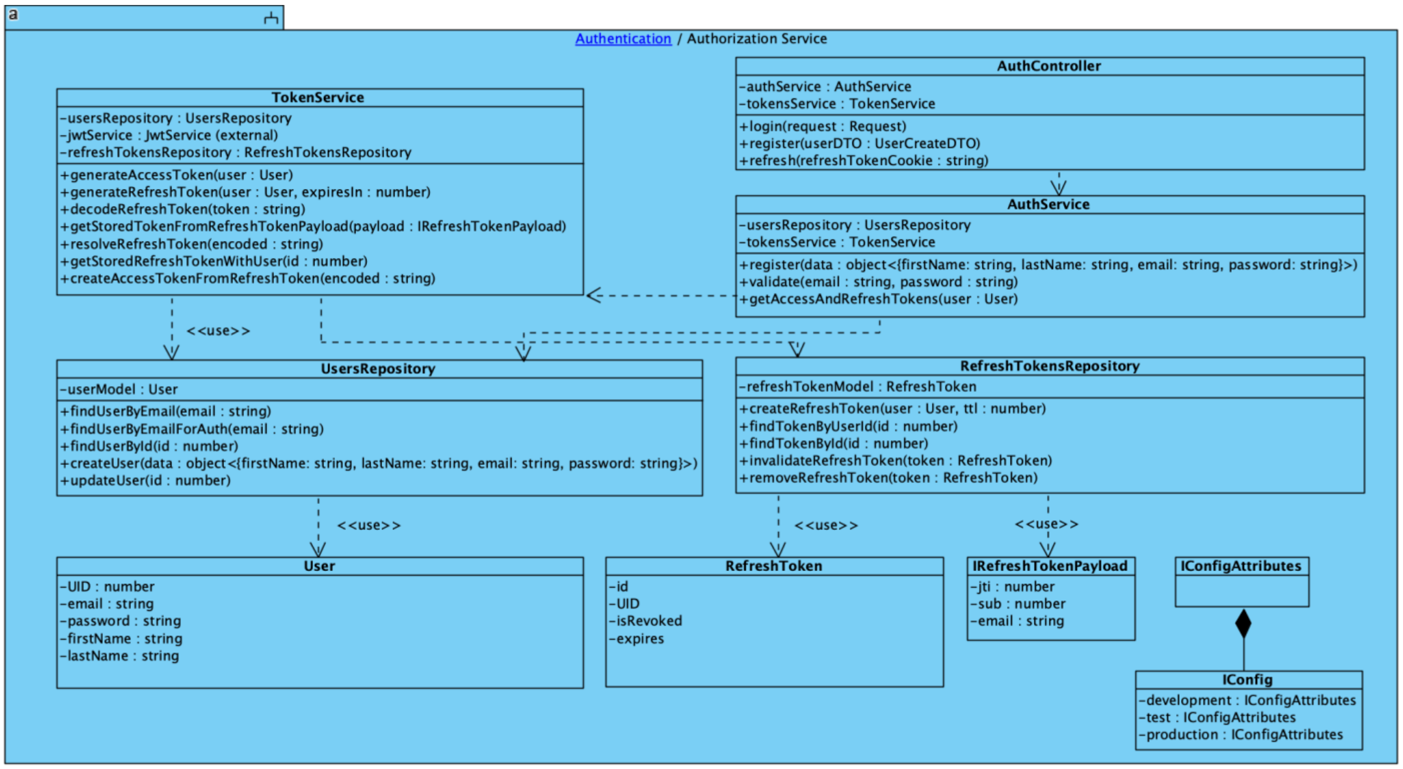

a
Authentication / Authorization Service
AuthController
-authService : AuthService
-tokensService : TokenService
+login(request : Request)
+register(userDTO : UserCreateDTO)
+refresh(refreshTokenCookie : string)
TokenService
-usersRepository : UsersRepository
-jwtService : JwtService (external)
-refreshTokensRepository : RefreshTokensRepository
+generateAccessToken(user : User)
+generateRefreshToken(user : User, expiresIn : number)
+decodeRefreshToken(token : string)
+getStoredTokenFromRefreshTokenPayload(payload : IRefreshTokenPayload)
+resolveRefreshToken(encoded : string)
+getStoredRefreshTokenWithUser(id : number)
+createAccessTokenFromRefreshToken(encoded : string)
AuthService
-usersRepository : UsersRepository
-tokensService : TokenService
+register(data : object<{firstName: string, lastName: string, email: string, password: string}>)
+validate(email : string, password : string)
+getAccessAndRefreshTokens(user : User)
<<use>>
UsersRepository
-userModel : User
+findUserByEmail(email : string)
+findUserByEmailForAuth(email : string)
+findUserById(id : number)
+createUser(data : object<{firstName: string, lastName: string, email: string, password: string}>)
+updateUser(id : number)
RefreshTokensRepository
-refreshTokenModel : RefreshToken
+createRefreshToken(user : User, ttl : number)
+findTokenByUserId(id : number)
+findTokenById(id : number)
+invalidateRefreshToken(token : RefreshToken)
+removeRefreshToken(token : RefreshToken)
<<use>>
<<use>>
<<use>>
User
-UID : number
-email : string
-password : string
-firstName : string
-lastName : string
RefreshToken
-id
-UID
-isRevoked
-expires
IRefreshTokenPayload
-jti : number
-sub : number
-email : string
IConfigAttributes
IConfig
-development : IConfigAttributes
-test : IConfigAttributes
-production : IConfigAttributes

Figure 6: Architecture diagram of authentication/authorization microservice

This figure depicts the main components and relationships within the Authentication/Authorization service available as a micro service within the cluster.

Figure 7: Architecture diagrams of additional microservices shows the smaller components within the rest of the larger components that can be seen as contained in the microservices cluster, including Neptune (with the elastic search service contained within) and Logstash.

Of course, the system would not be complete without proper access and storage in the system’s required data sources, given StudySnap’s data-driven nature. *Figure 8: Architecture diagrams of the data sources used* shows the significant data sources which help support the business logic of the system, which include the Elastic Search Index, and the StudySnap database.

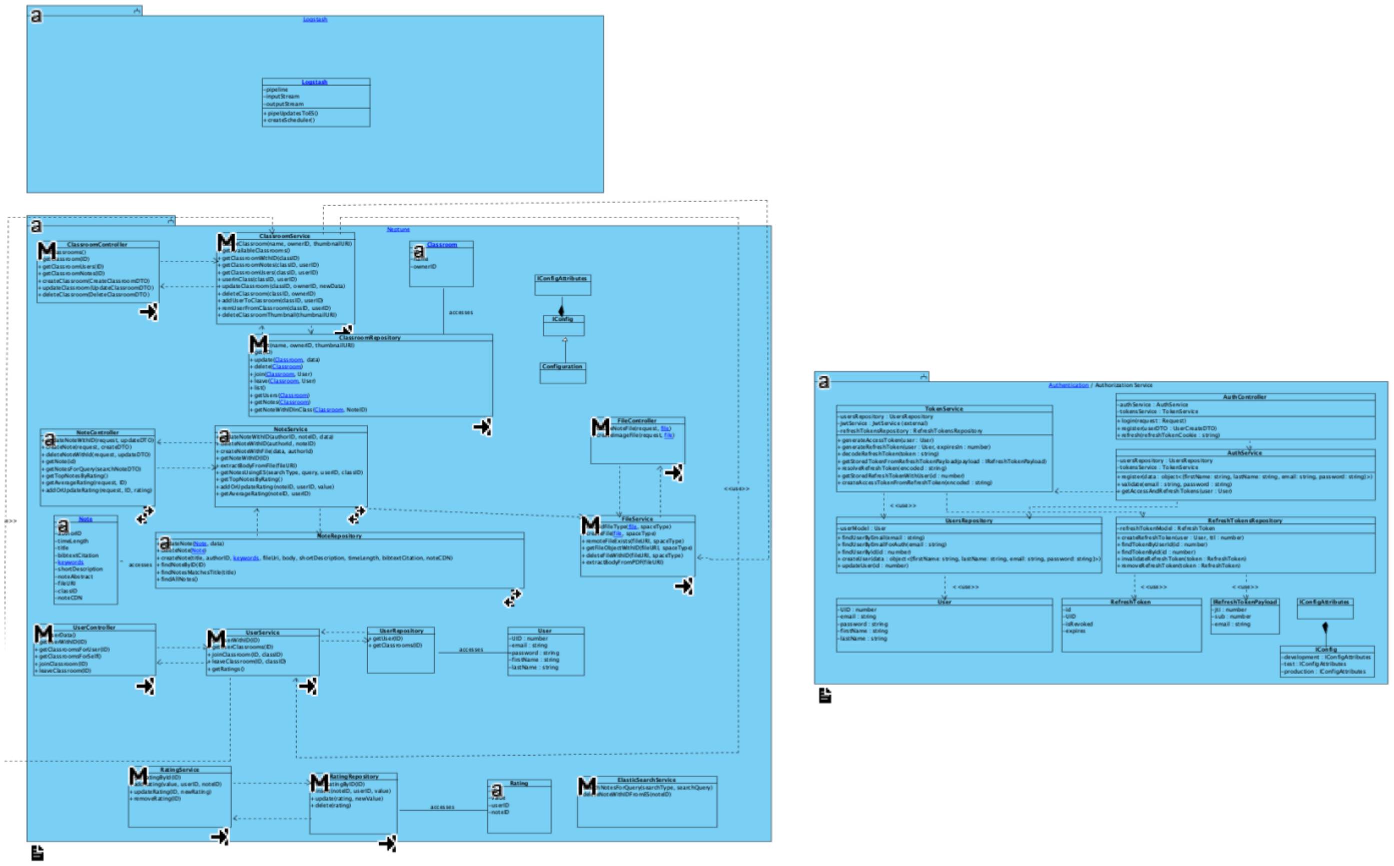


Figure 7: Architecture diagrams of additional microservices

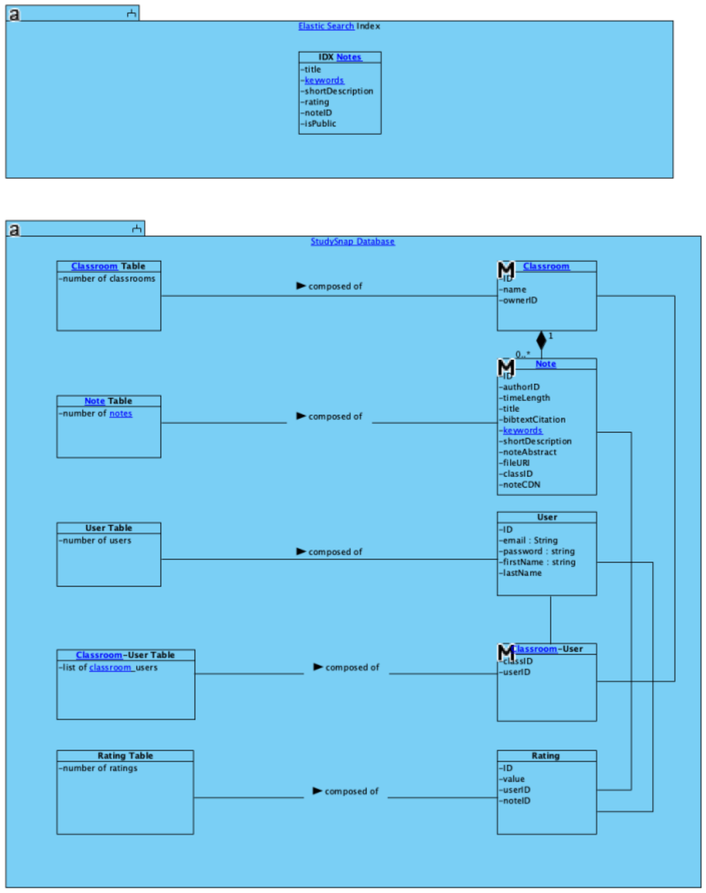


Figure 8: Architecture diagrams of the data sources used

As seen above in Figure 8, the main StudySnap database contains the tables for all of the main models used in the system. These include the classroom and the classroom table, the user and the user table, the rating and the rating table and the note and the note table. Each individual model-type is stored within its own designated table. The final model, the classroom-user, is used to map users to classrooms and classrooms to users. The classroom-user object contains both a classroom ID and a user ID. It holds these IDs to map a user to a classroom (the user being referenced by the held user ID) and a classroom to a user (the classroom being represented by the held classroom ID). The classroom-user table is used to hold the classroom-user mappings, in which both the classroom ID and user ID are used as foreign keys to represent the user and classrooms in their respective tables.

DEPLOYMENT MODEL

The use of the microservices architecture in the StudySnap system presents unique deployment challenges that would require further planning in order to make the system available to real users. The following deployment model details key steps and processes that are critical to getting the support systems online and client applications into the hands of users.

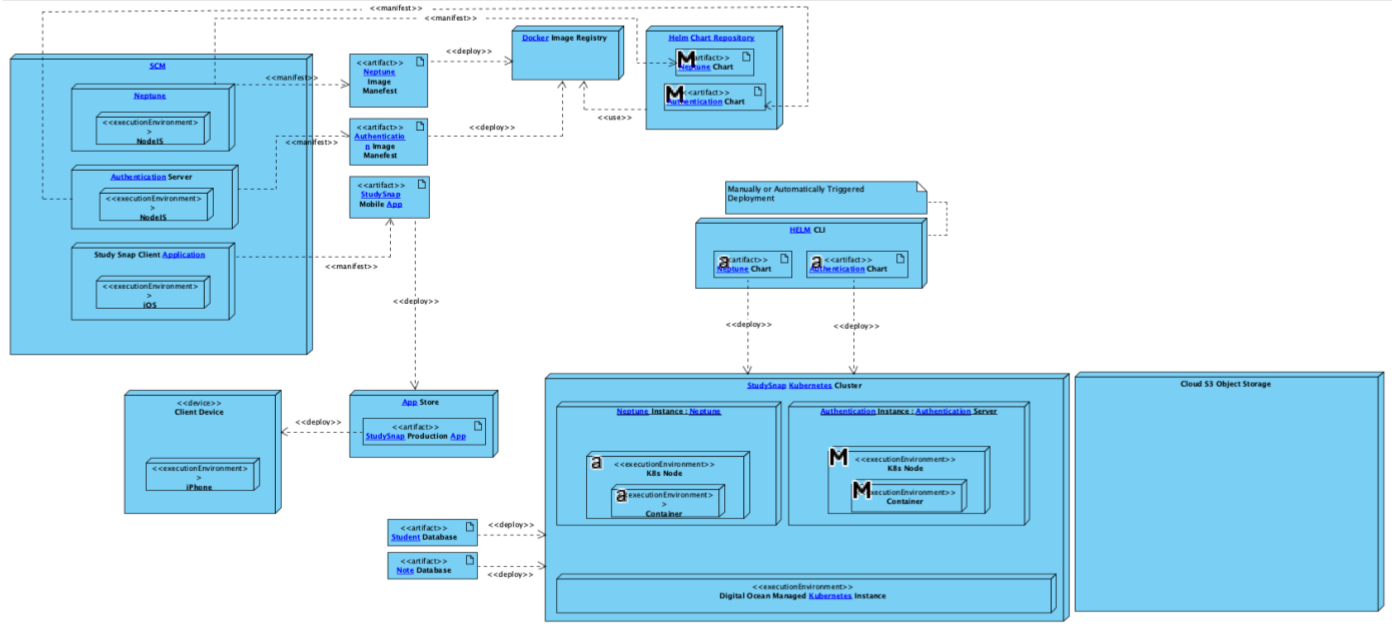

<<manifest>>
<<manifest>>
SCM
Neptune
<<executionEnvironment>> NodeJS
Authentication Server
<<executionEnvironment>> NodeJS
Study Snap Client Application
<<executionEnvironment>> iOS
<<manifest>>
<<artifact>> Neptune Image Manefest
<<manifest>>
<<artifact>> Authentication Image Manefest
<<artifact>> StudySnap Mobile App
<<manifest>>
<<deploy>>
Docker Image Registry
<<deploy>>
Helm Chart Repository
<<artifact>> Neptune Chart
<<artifact>> Authentication Chart
<<use>>
Manually or Automatically Triggered Deployment
HELM CLI
<<artifact>> Neptune Chart
<<artifact>> Authentication Chart
<<deploy>>
<<deploy>>
<<deploy>>
<<device>> Client Device
<<executionEnvironment>> iPhone
<<deploy>>
App Store
<<artifact>> StudySnap Production App
StudySnap Kubernetes Cluster
Neptune Instance : Neptune
<<executionEnvironment>> K8s Node
<<executionEnvironment>> Container
Authentication Instance : Authentication Server
<<executionEnvironment>> K8s Node
<<executionEnvironment>> Container
<<artifact>> Student Database
<<deploy>>
<<artifact>> Note Database
<<deploy>>
<<executionEnvironment>> Digital Ocean Managed Kubernetes Instance
Cloud S3 Object Storage

Figure 9: Deployment model of the StudySnap system

As seen in Figure 9: Deployment model of the StudySnap system, the process starts with the **application source code** and its **associated and supported execution environments**. These applications/services are then manifested into their **container image or compiled artifacts.** The container images are deployed to the **Docker image registry** (located at hub.docker.com). The **client application** is much simpler and is deployed to the app store where users can then download it freely. Continuing the deployment process for the containerized microservices, the deployment process is configured through **helm charts** that are stored safely in a private **helm chart repository**. From this point, the deployment to the **Kubernetes cluster** can be done either automatically, through CI/CD, or manually, which will host our system services. The triggered deployment will be performed using the **Helm CLI** either automatically through CI/CD or through a manual command. Installing the helm chart creates a release on the Kubernetes cluster that pulls and executes the containerized microservices and configures networking, storage, etc. for the deployed applications. One of the supporting systems that is not directly a part of the cluster is the Cloud S3 Object Storage, which is a managed service provided by DigitalOcean that still exists within the virtual private cloud. This allows untethered access to DigitalOcean's storage APIs, allowing quick access to upload and download notes and other data required by the application. It also provides a built-in content delivery network, which is utilized throughout the application to provide file streaming capabilities to the client application. Finally, once **pre-built charts for the databases** and other supporting applications are installed as dependencies to the existing microservices within the cluster, the deployment is considered complete.

PROJECT PLAN

In order to effectively plan our project out, several Jira features were utilized to lay out the work that was completed as well as the risks needed to keep in mind during development. Firstly, a Project Plan was created using Jira, which encompassed all the intended features, use-cases, stories, and releases. The project was divided up into several main categories and each use-case was assigned one of those categories for easy filtering. Furthermore, a few stories were included within each use-case to better understand the features that encompass each use-case. Beyond that, some developer sub-tasks were included within each story to identify the work that the developers needed to complete. These dev sub-tasks identify specific features of the application that were needed to satisfy each story and use-case. The 5 releases were also identified that were planned for development over the past year in their own section. For every release, stories were selected that needed to be completed over the sprint and were assigned equally to ensure all the work was done on time. These stories were then worked on and assigned time as the release continued. A link to said project plan can be found here: https://studysnap.atlassian.net/browse/SSPP. A Risk Management Plan has also been created, which allowed risks to be added and monitored as the project developed. See the Risk Management Plan section for more information.

ITERATION PLAN

As briefly mentioned above, sprint iterations are being tracked by the project plan created in Jira. This allows new sprints to be created under the format “Iteration 1 - …" which will identify the purpose of the sprint and how it affects the development of the project and the effect on the users. From here, stories can be added to each sprint as needed. As the project was developed, new use-cases and stories were created to support new developments. Please see below a list of the iterations as they were planned and completed:

Iteration 1 – This sprint will focus on setting up core infrastructure, developing wireframes and generic backend functionality, as well as preparing the PID for the inception release.

Iteration 2 – This sprint focused on the UI development, and project setup. We created Figma mockups of the UI and finalized the wireframes in Visual Paradigm. We also setup some of the project models in Visual Paradigm, which included creating the project glossary and the domain and requirements model. We also setup some of the backend as we saw fit.

Iteration 3 – In this sprint, we worked on setting up the architectural release of the project. This included creating the design model, interaction model and deployment model in Visual Paradigm, and creating the SwiftUI implementation for a few basic use-cases that showcase our architecture. We also worked on creating the functionality for these use-cases on the backend, and will implement them into the UI in the next release.

Iteration 4 - In this sprint, we worked on updating and finalizing our visual paradigm models and the PID. We also added the test model to Visual Paradigm to reflect the test-cases we had developed to validate that our implementation was working correctly. We also implemented the basic use-cases developed during past iterations into the front-end for demonstration.

Iteration 5 - In this sprint, we worked on integrating much of the feedback we obtained from the initial defense into both the software model, as well as the backend of our codebase. As we integrated some of the new features we had planned on adding, we continuously updated the design model, deployment model and interaction model to reflect the changes we had made.

Iteration 6 – In this iteration, we continued to add and finalize the new classroom features (namely, leaving and deleting classrooms), and updating the UI to reflect all of the new classroom functionality. Users can now join and create classrooms, upload notes to classrooms and search for notes inside classrooms. During the upload process, users can now use their phone's camera to take a picture of a physical note and use OCR to extract the text from the document into the application. Users then have the chance to edit the extracted text before it is added to the digital document. The note is then uploaded as normal.

Iteration 7 - In this iteration, we added many additional features that we had been planning on adding previously. We also made a change to the backend's architecture, and adjusted the authentication flow. Features that were added in this iteration included rating the note, adding citation information, viewing citation information, viewing the profile, the ability to change your password, adding thumbnails to the classrooms and viewing your own personal notes in a separate storage area.

Iteration 8 - In this final iteration, we focused on adding a few small additional features, such as adding the terms and services confirmation, the total number of ratings for each note, and a few UI fixes. We also focused on fixing some small bugs on the backend, and deployed the project to the Kubernetes cluster for demonstration. This iteration was also used to finalize all the core features and prepare the application for the defence.

RISK MANAGEMENT PLAN

One of the main components of the project plan not yet described is the risk management plan. This was also created using Jira and can be found at this link: https://studysnap.atlassian.net/browse/SSRMP. The purpose of the risk management plan is to identify risks that are associated with the development or deployment of the project and keep track of them throughout. If risks and their possible effects on the project can be identified well before they happen, there is greater opportunity to mitigate them. Like the stories, new risks were identified as progress on the project was made, so it was imperative to keep track of both new risks and old ones. As of the initial risk management plan creation, basic risks have been identified, such as "work conflicts with other courses or responsibilities" and "difficulties using new resources and technologies", but more werecadded as development continued. Below is a list of the top-5 risks that were initially identified and a brief description for each:

Difficulty in using new tools or technologies - Due to our lack of experience with some technologies, some implementations may either take longer than expected or won't work at all.

Cost of available cloud resources becomes too much - In order to keep development of the project going, the cost needed to keep running the cloud could become more than is available.

Cloud resources no longer sufficient for project requirements - We realize part-way through that our current cloud provider doesn't do everything we need, causing us to switch cloud providers.

Schedule conflicts with other coursework - Other course work gets in the way of project progress.

Change in direction or process while project is being developed - We realize that some previously intended features no longer fit our main feature list, causing us to rethink certain developments.

POST-INCEPTION DEFENSE PROJECT PLAN CHANGES

Following the initial defense of the project, several new changes were brainstormed and planned to be integrated throughout the following semester. The core of these changes involves the integration of the new classroom feature, a way to limit the context of user note sharing and user interaction into logical groupings based on real-life courses. Ideally, users will be able to create or join a classroom in the application for every actual course they are enrolled in, and then share and view notes related to each course in each respective classroom. Of course, making such a change will involve many changes to the previous architecture, which have been identified in the project plan.

Such tasks that involve making these changes include making updates to the different UIs that involve classroom functionality, updating the Visual Paradigm models to reflect the proposed changes to the architecture, making changes to Neptune to handle the new classroom model, as well as updating the database schemas to reflect the new models for notes, classrooms and users. Of course, additional tasks have also been identified related to fixing previous errors or bugs left over from earlier in the project. These bug fixes and additional classroom features have all been grouped together into iteration 5, however, it is possible that some tasks will move into iteration 6. The goal is to have the classroom feature fully integrated by the end of iteration 5, so that iteration 6 can focus on adding additional core features not yet present. Such core features include note uploading and OCR integration. Iteration 7 will involve integrating the classroom feature into the note searching functionality, and ensuring the note search works correctly. Iteration 8 will focus on adding additional user management tasks, like adding a logout feature and a change password feature. As user management is not a core feature, it is possible these tasks will be moved into a later iteration if other necessary tasks come up.

As seen in the Interaction model of the project, StudySnap has been broken up into 3 logical use-case areas: note organization, note upload and viewing and study material provisioning. Malik Sheharyaar Talhat has been assigned use-cases associated with note organization (sorting notes, displaying notes, moving notes, etc.). Liam Stickney has been assigned use-cases associated with note uploading and viewing (uploading notes, viewing notes, physical to digital note conversion, etc). Ben Sykes has been assigned use-cases associated with study material provisioning (note searching, quiz generation research, etc.). It is likely that the new classroom feature will contain use-cases relevant to all three functional areas, so these areas will be monitored as development of the classroom feature continues.

Finally, the risk management plan has been updated slightly to include updates to the risks previously identified based on the new classroom feature. One of the risks previously identified was related to changing direction of the project, which the new classroom feature has invoked. The mitigation strategy for said risk was followed, that being brainstorming new ideas related to the change in direction, and coming up with solutions that could be integrated at manageable cost. Several comments have been added to previously identified risks, and more will be added as development of the project continues.

PROJECT RELEASES

Five main releases have been identified that will be completed over the project's lifetime. As written in the project plan on Jira, they are as follows:

Inception Release - The initial release of the system, which is UI driven and visualizes the core features of the applications.

Elaboration Release - The release of the system which contains a fully working UI and includes some of the intended core functionalities.

Alpha Release - The release following the elaboration release, includes most of the core features and potential bug fixes/UI improvements.

Beta Release - The release following the alpha release, will also contain additional features, or fixes to the previously added features in the alpha release.

Final Release - The final release of the system. Will include all the intended extra features and will be tested to fix any outstanding bugs or issues.

RESPONSIBILITY MATRIX

Responsibility	Ben Sykes	Liam Stickney	Malik Sheharyaar Talhat
		PROJECT MANAGEMENT	
Project Owner	NO	YES	NO
SCRUM Master	YES	YES	NO
Risk Analyst	NO	NO	YES
		REQUIREMENTS ENGINEERING	
Requirements / Business Analyst	YES	NO	NO
Stakeholder Champion	NO	YES	NO
Functional Area Champion	NO	YES	NO
User Experience Design Lead	NO	NO	YES
		SOFTWARE ARCHITECTURE	
Software Architect	NO	YES	NO
Requirements Model Lead	YES	YES	YES
Domain Model Lead	YES	YES	YES

Design Model Lead	YES	YES	YES
Deployment Model Lead	YES	NO	NO
Interaction Model Lead	YES	YES	YES
		CONSTRUCTION	
Full-Stack Developer	YES	YES	YES
Integration / DevOps	YES	NO	NO
		TESTING	
QA Lead	NO	YES	NO
Verification & Validation	NO	NO	YES
Test Model Lead	YES	NO	NO
		SUPPORT	
Tool and Devices Support	NO	NO	YES
Communication Support	NO	YES	NO

VALIDATION AND TESTING

This section briefly describes how the testing strategy has been designed and implemented and the analysis performed on the test results in order to determine whether the system is functioning as intended throughout development. Issues on GitHub were used for some of the more immediate defects that appear in the associated code repository. This can be found at the following link: https://github.com/Study-Snap (**Note**: Many of the StudySnap source code repositories are private and will require explicit permission to access. The supervisor has access currently and can provide further access upon request).

TESTING STRATEGY

In order to validate functionality of system requirements, the team decided to take advantage of Github Actions CI/CD in order to perform automatic unit and e2e testing on the system components. This includes, but is not limited to, the SwiftUI iOS application and associated microservices (Neptune and Authentication + DB services). Since the project uses continuous integration, the systems were continuously developed, and test releases were created automatically. The only requirement is that for each new feature or component added to the system, any required tests are added to their associated testing suites. Inside the CI process, the system takes advantage of Jest(https://jestjs.io/docs/getting-started) and Supertest (https://www.npmjs.com/package/supertest) to perform Unit and end-to-end testing on the system.

The testing/validation process also involves the use of Linting software such as EsLint (for NodeJS, https://eslint.org) and SwiftLint(For SwiftUI linting, https://github.com/realm/SwiftLint). This maintains high levels of code quality and avoid code smells with each commit to the source code repositories.

The Requirements of the testing process (including test suites and procedures) are included in the testing model shown in Figure 10, Figure 11 and Figure 12 below. These figures visualize our test plans detailing when and how test-cases will be applied to each functional area. For more details see the VPository.

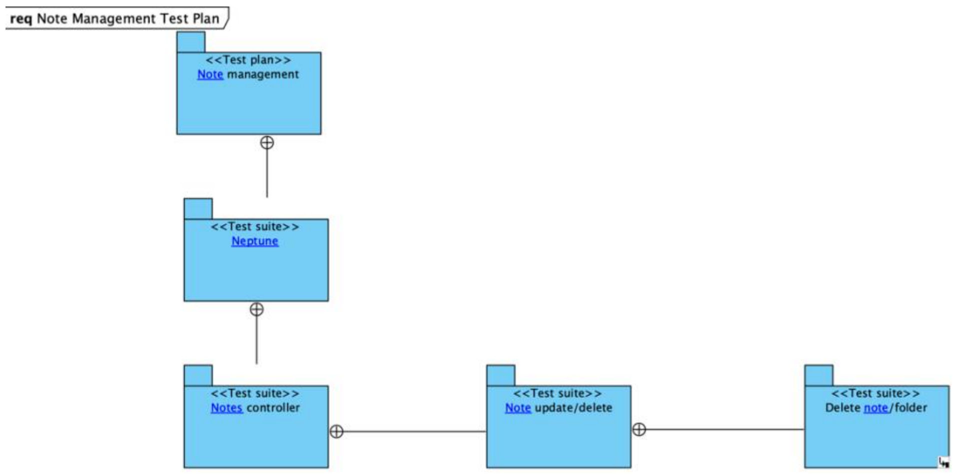

req Note Management Test Plan
<<Test plan>> Note management
<<Test suite>> Neptune
<<Test suite>> Notes controller
<<Test suite>> Note update/delete
<<Test suite>> Delete note/folder

Figure 10: FA1 Note Management test plan

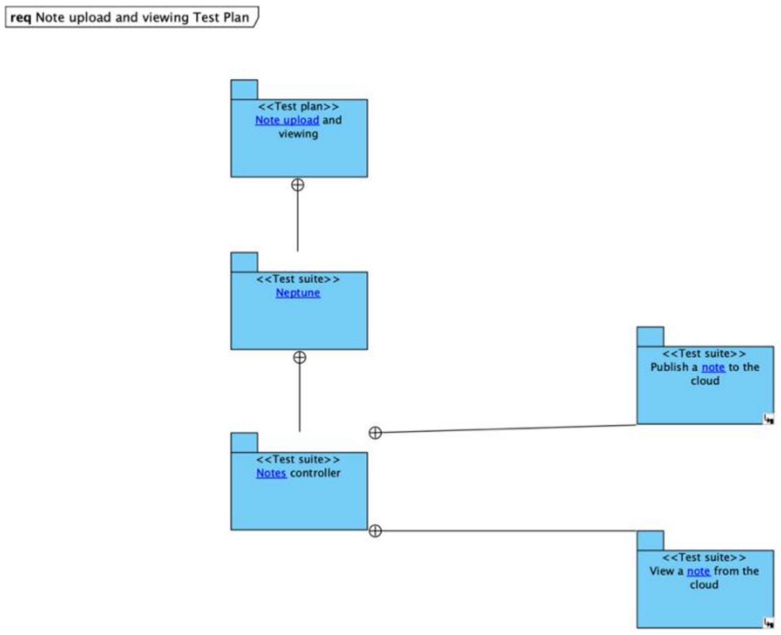

req Note upload and viewing Test Plan
<<Test plan>> Note upload and viewing
<<Test suite>> Neptune
<<Test suite>> Notes controller
<<Test suite>> Publish a note to the cloud
<<Test suite>> View a note from the cloud

Figure 11: FA2 Note upload and viewing test plan

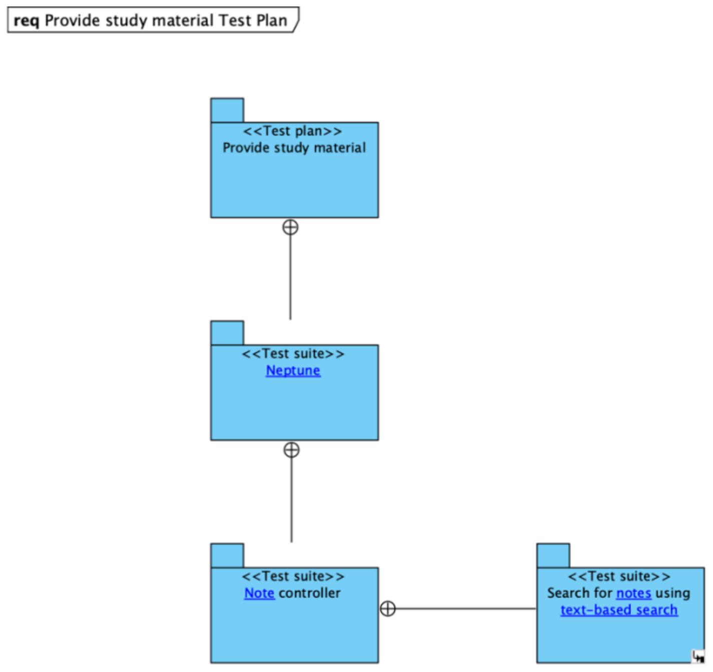


Figure 12: FA3 provide study material test plan

VALIDATION RESULTS

At the time just before project completion, there are no significant issues to speak of present in the application. As testing continues prior to the final submission, a few small UI issues are present here and there, but they do not pose a significant threat to the overall operation of the application. All significant back-end issues have been fixed, namely due to the transition to object-based storage, which fixed many of the previous back-end issues. The system is very stable at this stage, due in part to the quality of testing done throughout development. This was done through both manual testing, done primarily to test back-end integration on the front-end, as well as automated testing, which was performed on all major functionality on the back-end. As new features were added on the back-end, automated tests were created to validate the features and ensure they did not break already existing features. Furthermore, a test-driven development environment was enforced on the back-end development, in which test-cases were written for each feature before code was created for each feature. Once the test-cases were put into place, code was then written to satisfy each test-case, rather than satisfy the features themselves. This helped ensure that features were only added to the back-end after it was validated that the feature would pass all tests and not break any pre-existing features.

Additionally, based on the user tests conducted on the application, the solution constructed runs very well when deployed in its intended environment. The application was free of any crashes or major bugs, and the different features of the application ran in a good amount of time. The majority of feedback for the application from the user tests was focused on the front-end, and where certain features were located. Once every feature was located and executed, the system behaved as intended. The users who conducted the testing were appreciative that many features were easy to access and not overly cumbersome, and specifically mentioned that every feature in the application seemed to execute well.

CONCLUSION

The problem that is being solved by the system is the unorganized and unprepared lifestyle that students face when working through a cluttered school environment. School, whether it be high school or college/university, can be daunting for the unprepared, whether it be because of the number of papers and notes the student must keep track of, the stress of cramming for an upcoming due date, or the uncertainty of the preparedness for an upcoming test or exam. Traditionally, students have faced problems related to the organization of many papers or notes, finding relevant study material to aid in preparing for an upcoming test, and trying to find a way to validate one's knowledge of a certain topic or field of interest. StudySnap provides several useful features that solve these problems. The first main feature being an easy-to-use **note organization system,** which allows users to upload their notes into **logically organized classrooms**, which provide users with an easy-to-use note organization tool. The second main feature is a semantic search-based **note search system,** which allows users to enter in a query or set of queries, and the system will return a list of "hits" based on their relevancy to the inputted queries, ensuring that these hits are relevant to what the user wants to study in that moment. These hits are comprised of user-uploaded notes and allow users to share their notes with others to create a user base of collaboration-driven students. Finally, through the aforementioned classrooms, users can rest-easy knowing the notes they are searching for are directly related to what they've personally learned in class, since the notes uploaded within each classroom can only be uploaded by users the user has directly invited. This encourages a collaboration-based learning experience, in which users can learn from each other's uploaded content.

PROJECT SUITABILITY

StudySnap is suitable for deployment in the targeted domain as it is was built to specifically solve the problems not currently solved by other existing applications. Firstly, as described in the Project Overview section, there is a clear problem among students that correlates poor mental health with poor academic performance. This suggests a latent need for some sort of solution that relieves the stresses related to said academic performance, hence the popularity and number of pre-existing related solutions already present in the domain. Of course, it is unrealistic to expect that a set of study-helper applications will fully solve these complex issues, however, it is hoped that they can at least begin to improve the quality of students' academic performance, which may be a key step in relieving the stress and anxiety that creates these poor mental health problems. As mentioned, there are several applications that do already exist on the market that are related to StudySnap. However, many of these applications do not fully solve the problems that StudySnap aims to solve. For instance, an application such as Chegg allows for the uploading of notes to share with other students, where users are then able to access these notes for additional study. However, besides a simple manual search, the user has no real way of being able to tell if these notes are going to be remotely related to what they want to study. Considering these manual searches are typically based on the titles of the documents, it is a possibility that a student clicks on a note with a title that *seems* like it'll contain information they want to see, where in reality it is almost completely unrelated. **StudySnap remedies this problem with a search engine that allows users to search through the full text of the content of each note, rather than just a title, so they can ensure that the note's contents actually contain relevant information to their query.** This advanced search feature will allow users to gain more control over what content they view and provides a unique solution to the problem not fully addressed by other existing applications.

An additional feature that came up after the initial project defense was the idea of creating classrooms, which act as logical groupings that sort the notes in the database. Essentially, a user can create a classroom, which acts as a digital version of a class they are currently enrolled in in real life, and they can then invite other users by using a uniquely generated invite code. The user, and any invited users, can then upload their notes within the classroom, where users can then search for them using the above-mentioned search feature. This not only **encourages students to collaborate with each other**, but also ensures that the notes the user is currently searching through were uploaded by people in the same class as them. This helps **to ensure that the content they search for is actually related to what they've learned in class.** Many other applications on the market seem to focus on the learning of the individual, rather than the learning of an entire class, or group of students. StudySnap breaks the mold in this case, as it actively encourages students to share their notes with their classmates, which creates a mutually beneficial learning environment.

Of course, it was acknowledged that simply agreeing upon features that seemed useful was not enough to validate the decision. As such, a domain expert needed to be identified to confirm that these features would actually be relevant if deployed in the real-world domain. The domain expert we consulted for this reason was Jamie Goodfellow, the Manager of Learning Services at Sheridan College. As the manager of Learning Services, Jamie has a lot of experience in interacting with students and understanding their academic needs when concerned with studying and academic assistance, including improving time management skills, note taking skills, studying and test management skills and tutoring, all of which relate to skills and features that StudySnap aims to provide [3]. Furthermore, Jamie has direct contact with other employees within the Learning Services office, including learning strategists and tutoring experts, and was kind enough to incorporate some of their feedback in her evaluations of the proposed solution as well. Overall, Jamie indicated that she felt that the features we suggested would be very helpful if deployed in this domain. Specifically, Jamie indicated interest in both the note organization feature and the note search feature. On the note organization feature, Jamie said "I like this idea! It's a great idea to have notes stored in one space and the search functionality is great. It's also helpful to teach students that organization of notes is key (in addition to taking good notes!)". However, she also indicated possible red flags related to academic integrity and copyright infringement when sharing notes with other students. As it was felt that sharing notes was a key feature in creating the proposed solution, a remedy to the possible academic integrity breach was created with the help of the project supervisor, and Jamie confirmed that this remedy created was sufficient in mitigating the risks associated with cheating. This process is explained further in the following section. With this issue settled, it was agreed upon that the features proposed for StudySnap's deployment would be suitable for providing users with a unique solution not present in the domain.

In terms of the feasibility of StudySnap's construction, it was decided to develop StudySnap as an iOS application as iOS is a platform that StudySnap's creators had experience using and were comfortable doing so. Developing the solution as a mobile application also made logical sense, as it allowed users to utilize its features on-the-go. For instance, users can open the application to quickly read through their notes on a crowded train or bus, something that is a lot less feasible on a laptop or computer. During the first few weeks of development, it wasn't completely clear how certain developments rooted in advanced areas of computer science would go, namely areas related to cognitive computing and artificial intelligence, due to a lack of experience. However, through supplementary courses in these areas, there have been opportunities for significant research and development, and features based on these areas like the semantic-search and optical character recognition were then confidently implemented into the solution. StudySnap's adoptability into the domain is then a rather straightforward one, as the prime distribution medium of these “study helper" applications is through the App Store or Google Play Store, where thousands of users can easily access the application. From there, users just need to download and run the application to access its unique features.

DOMAIN EXPERT EVALUATION

As mentioned in the previous section, the domain expert that was contacted for the project's development was Jamie Goodfellow, Manager of Learning Services at Sheridan College. In order to determine the suitability of the features that had been planned, the features needed to be evaluated by somebody who not only worked in and around the domain of education and student assistance, but also someone who had direct contact with students and those who assist students directly. That way, the evaluations could not only factor in the expert's personal opinions, but also the opinions of students themselves, of which the expert has likely come into contact with on numerous occasions.

With this in mind, Jamie Goodfellow was contacted to provide her thoughts and opinions on StudySnap's main features. As Manager of Learning Services, Jamie not only has experience in providing students with time management, note taking and study preparation advice, but also has direct contact with other experts, including Learning Strategists and tutoring experts. Jamie was kind enough to also speak with these other experts and factor in their thoughts into her feedback, so the feedback received was that of several domain experts. What needed to be evaluated then, was the feasibility of the features that were being planned for development. Specifically, how much students would actually make use of the main features that were planned, those being the note organization system, the note upload and sharing on the cloud, and the note searching. Jamie's initial feedback, as alluded to above, was that she felt that the note organization system tool was very interesting and would encourage students to keep their notes organized and take thorough notes. However, she had concerns with both the note upload system and the note-search system (which are related). Specifically, she was concerned with potential for an academic integrity breach, considering the idea of sharing notes could encourage students to cheat on assignments or tests. These were risks that were initially identified during the solution's original inception but hadn't been given additional thought up until this point. As it was agreed upon by the project team that the note upload and search systems were integral to the solution's development, a mitigation system needed to be put in place to minimize the risks associated with cheating and copyright infringement.

During discussions with the project supervisor, it was decided that completely reducing the chance of cheating or copyright infringement to an absolute zero wasn't feasible for the project's current developments, and likely out of scope. However, three strategies were identified that could be put into place to mitigate the risks as best as possible.

The first of these three strategies involved creating a terms and services document that specifically identify the rules and what users are and are not allowed to do while using the application. These rules would include not uploading stolen or uncited content, not uploading assignments or test documents, etc. Users are then shown the document upon signing up, where they would then need to check a box confirming their conformation to the document. The second strategy created was to only allow the uploading of personally created content, that is, notes written entirely in the student's own words. As stated on Sheridan's website concerning sharing notes on course-sharing websites "if you take your own lecture notes (not verbatim or word-for-word of what your professor said), you own copyright to your notes and can share them with other students" [4]. This rule would be clearly defined in the terms and conditions document. The final strategy would be to allow users to cite their notes when they upload them. As mentioned, students who upload their own lecture notes in their own words own the copyright to these notes, and do not actually *have* to cite them. However, adding the citation option will allow students who feel as though they should be referencing their professor or any other source to do so and therefore lower the risk of a copyright infringement even further. The Academic Integrity Office at Sheridan College was also contacted, who confirmed that citing a professor's name in this way is allowed, such that the student has permission from the professor. With these strategies in mind, Jamie was updated on these new developments for her opinion. She was pleased with these strategies being put in place and appreciated the engagement in the Academic Integrity policy. She also noted that the citation feature was additionally beneficial as it makes the citing process easier for students while encouraging them to cite. With these new strategies put in place, Jamie felt that the features planned were suitable for development and any potential risks were identified and subsequently mitigated.

USER TESTIMONIALS

Based on the user testing that was conducted prior to the final release of the system, the application works very well when deployed in its intended environment. In every test conducted, the application remained stable while operating in the hands of users, and no crashes or major bugs were reported. Some of the user feedback obtained highlighted the fact that the application ran quite quickly, particularly when running on an actual physical device as opposed to an emulator. The majority of the feedback focused mainly on the front-end of the application. Some users expressed that particular features, like the delete note option, were not necessarily easy to access, or not very clear on where to access them. One user also commented on the log-in page and the profile view page, noting that both pages felt very barren and lacking compared to other screens of the application. To adjust for this feedback, both the sign-up page and log-in page have been changed to give the application more of a "business-casual" feel. Furthermore, the profile view was updated to not only have updated visuals, but to also include more user information, as it not only makes the profile view less barren, but also provides additional information that the user may find useful. When tested in the hands of an actual user, it was also confirmed that StudySnap integrates well with built-in and third-party iOS functionality, such as password managers, the camera, and file and photo storage.

FUTURE WORK

Beyond the implementation of the features identified in this document, there are several possible additional areas of functionality that could improve the solutions usability and functionality. Firstly, as identified by the domain expert, several mitigation strategies need to be put into place to mitigate the risk of an academic integrity breach and copyright infringement. One of the strategies identified included the addition of a terms and services document, which would identify the rules and regulations that users must follow while using the application. Beyond just a simple document outlining some rules, it is unlikely that, during this first year of development, a full-fledged terms and services document could be created without some sort of legal assistance from a third-party source. Obtaining this legal assistance would likely cost money that is not readily available to be spent, especially over the coming year. However, creating this document would be integral in mitigating the risks identified earlier, so in during the future work of this project, the correct funding and legal advice would need to be obtained to deploy this application in its intended domain without risk of copyright troubles.

In order to target additional stakeholders, there could be additional work done on the login and account system. Specifically, a separate account system could be created for teachers to separate them from students. Teacher accounts could allow instructors to upload private content specifically for the students they are currently teaching, and monitor progress made by viewing the reading habits of their students and testing their knowledge by posting frequent quizzes and assignment questions. Currently, as all the identified requirements fall under the use of students, this addition was out of scope of the current year. However, its inclusion at a later date would allow the solution to target an entirely new user base of instructors and encourage greater academic collaboration between students and teachers.

A web-application companion would also work very well in support of the mobile solution developed. The ability to upload and access the same content on a laptop or computer, for instance, would work very well in tandem with the mobile application, and would help in targeting additional users and make the application more accessible. While it is likely that most of the intended users would have a mobile device, it is not so likely that every intended user would have a desktop computer or laptop, so keeping the mobile application the primary mode of delivery of the solution ensures the most end-users are targeted. That being said, a web-application companion would increase usability and would be a welcome later addition to the solution.

In terms of work done on the actual functionality of the project, there are several areas for improvement. One of which involves the quiz generation tool. The most likely scenario would be that the user enters in his own questions, and the system finds the answers for him, and then generates a corresponding quiz. Services such as Microsoft Azure offer these cognitive question and answer capabilities [5]. However, an ideal quiz generation tool would be one that not only generates answers, but generates the questions themselves, by identifying statements made in the document and pulling information out of them. The feasibility for such a feature is currently unknown, however, and its inclusion will be saved for a future date. However, considering the helpfulness of such a feature, an initial solution has been developed as a proof-of-concept solution, and research has been put into the idea to confirm that it would be both feasible and helpful considering the current implementation.

Another helpful addition could be made to the note search engine, to include external sources beyond just uploaded notes. Currently, the search feature just iterates over user-uploaded notes on the application. However, if this could be expanded to also include resources found on the internet, it would definitely increase the power of the searching capabilities. Doing so would likely involve many formatting and possible copyright issues, however, so it is likely that this addition would need to be made beyond this first year.

A main use-case that was initially considered during the first month of development, and which slowly morphed into the note search engine was the idea of a note recommendation engine. The idea would be to have a "card-based" system, in which users are provided with several note "cards", where they can swipe right or left to indicate their interest in said note. The cards shown to the user could be based on their own personal reading habits. If every note is annotated with some keywords, the system could keep track the most common keywords the user spends time viewing, or the keywords found most frequently in that user's note storage. This would allow for the inclusion of some data analytics technologies that could be used to keep track of a user's interests in the background. It could even display weekly or monthly statistics related to the reading habits of users. Based on these statistics, the search engine could perform a search in the background without user input based on that user's reading habits and display the most relevant notes related to these statistics in the aforementioned card view. This particular feature would be useful as it further automates the finding of study material and means users can obtain additional relevant study material by simply using the application.

BIBLIOGRAPHY

[1] Canadian Industry Statistics. (2019, March 01). Retrieved February 11, 2021, from https://www.ic.gc.ca/app/scr/app/cis/search-recherche?lang=eng

[2] Orozco, R., Benjet, C., Borges, G., Moneta Arce, M., Ito, D., Fleiz, C., & Villatoro, J. (2018, January 24). Association between attempted suicide and academic performance INDICATORS among middle and high school students in Mexico: Results from a national survey. Retrieved February 11, 2021, from https://capmh.biomedcentral.com/articles/10.1186/s13034-018-0215-6

[3] "Academic & Research Skills," *Sheridan Library and Learning Services*. [Online]. Available: https://sheridancollege.libguides.com/sb.php?subject_id=30820. [Accessed: 11-Apr-2021].

[4] "Copyright Services Guide: Course-Sharing Websites," *Sheridan Library and Learning Services*, 20-Jan-2021. [Online]. Available: https://sheridancollege.libguides.com/copyright/coursesharing. [Accessed: 11-Apr-2021].

[5] "QnA Maker API: Microsoft Azure," *API | Microsoft Azure*. [Online]. Available: https://azure.microsoft.com/en-ca/services/cognitive-services/qna-maker/#features. [Accessed: 11-Apr-2021].

Technologies Used

"Product Docs Home," *DigitalOcean Documentation*, 26-Jun-2018. [Online]. Available: https://docs.digitalocean.com/products/. [Accessed: 13-Apr-2021].

"Kubernetes Documentation," *Kubernetes*, 12-Nov-2020. [Online]. Available: https://kubernetes.io/docs/home/. [Accessed: 13-Apr-2021].

"SwiftUI Framework," *Apple Developer Documentation*, 2021. [Online]. Available: https://developer.apple.com/documentation/swiftui/. [Accessed: 13-Apr-2021].

"Azure documentation," *Microsoft Docs*, 2021. [Online]. Available: https://docs.microsoft.com/en-us/azure/?product=featured. [Accessed: 13-Apr-2021].

"GitHub Documentation," *GitHub Docs*, 2021. [Online]. Available: https://docs.github.com/en. [Accessed: 13-Apr-2021].

"Visual Paradigm Documentations," *Visual Paradigm*, 2020. [Online]. Available: https://www.visual-paradigm.com/support/documents/. [Accessed: 13-Apr-2021].

"Jira Documentation," *Atlassian Support*, 2021. [Online]. Available: https://confluence.atlassian.com/jira/jira-documentation-1556.html. [Accessed: 13-Apr-2021].

H. Xiao, "bert-as-service Documentation¶," *bert-as-service*, 2018. [Online]. Available: https://bert-as-service.readthedocs.io/en/latest/. [Accessed: 13-Apr-2021].

"Elasticsearch Guide," *Elastic*, 2021. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html. [Accessed: 13-Apr-2021].

"Logstash Reference," *Elastic*, 2021. [Online]. Available: https://www.elastic.co/guide/en/logstash/current/index.html. [Accessed: 13-Apr-2021].

K. Mysliwiec, "Documentation: NestJS - A progressive Node.js framework," *NestJS*, 2021. [Online]. Available: https://docs.nestjs.com/. [Accessed: 13-Apr-2021].

"PostgresSQL: Documentation," *PostgreSQL*, 2021. [Online]. Available: https://www.postgresql.org/docs/. [Accessed: 13-Apr-2021].

Postman, "Postman Introduction," *Postman Learning Center*, 2021. [Online]. Available: https://learning.postman.com/docs/getting-started/introduction/. [Accessed: 17-Apr-2021].

APPENDIX

StudySnap Software Requirements Specification (SRS) Document Link: https://sheridanc-my.sharepoint.com/:w:/g/personal/stickney_shernet_sheridancollege_ca/EaNlfTfGfr9LmIoVz7LMawYBlXAaGD5WsXDXPNo0S-s07w?e=dQxoZz

StudySnap Software Design Document (SDD) Link: https://sheridanc-my.sharepoint.com/:w:/g/personal/stickney_shernet_sheridancollege_ca/ERbgCBgBV6ZHs1t6QP-vB_0BhvRg6PlZRgcMVcsoXglU3Q?e=CcvmSY