1. Introduction

Overview

the cloud with the assurance that their data is stored securely and safely on the premises of DCU.

We are currently proposing two clients, one for the desktop to synchronise files, and another on the web where a user may view, add, and remove files of their choosing. Given that we cannot have access to DCU's servers to install this ourselves, we are going to include an installation process that would assist a server admin in setting up the application. This will not hinder our project itself, as a proof of concept will still be presented by substituting DCU's server with another machine.

Need for the system.

Our project idea evolved from the realisation of the dependance of the University on Third-Party applications (such as Google Drive and Microsoft One Drive). With the rise of Third-Party authentication, cloud solutions, and other applications there is an often overlooked fact that we (as consumers) are trusting our identities and our data with companies that we have no reason to trust. These companies have access to our data, and may view things that we wish to keep private. Furthermore, we are not in control of the availability of our data. Whether or not our data is there tomorrow, or our accounts for that matter is in control of the companies that we are entrusting with our data. This is what led us to the idea of XDrive.

The XDrive will act as a substitute for these Third-Party applications for students of the University. It will be a developed storage solution that will fulfil the same requirements as the aforementioned products without the caveat of trusting a Third-Party entity. This will ensure privacy and bypass the terms & conditions set by the companies mentioned above, to ensure that the data which originates from a student/staff of the university, resides and abides by the terms and conditions of the University.

Interactions with other systems.

As mentioned previously, the application will offer two clients. One for the desktop and the other for the web. The desktop client will enable automatic synchronisation

of student files to ensure minimal data loss. The web client, on the other hand, will allow students to add or delete files of their choice. Synchronised files will be viewable on both the desktop and web interface respectively. Depending on time, an option will exist to add mobile phone access in which users will have access to their files using their mobile devices.

Business Context

This product will be used mainly by educational institutions. However, it can also be used by organisations to store their files, documents and folders under a private cloud platform that is controlled by the system administrators of their company. This will cement trust and privacy in the office or educational institution while also allowing flexibility on the term and conditions. This will be because the system administrators at each organisation will remain in control of the files their student/staff have created and can apply policies, such as limitations on what, with whom, and how files can be shared and How much space can be allocated to each member.

Glossary

API - Stands for Application Program Interface. It is used to communicate with servers to get, post or delete data. Synchronisation - “Synchronisation is the precise coordination of multiple events or mechanical devices. In computing, it refers to the coordination of hardware devices, such that the data they contain or provide is made to be identical. The synchronisation is usually done within an acceptably brief period.” (acknowledgement to computerhope.com) Third-Party Application - “A third-party app is a software application that was developed by someone (a person, or a company) other than the original vendor of the platform that the app was created for. Examples of third-party apps include OS X and iOS software not developed by Apple. Android software not developed by Google.” (acknowledgement to quora.com) Desktop Client - In this document, the desktop client refers to the pre-installed version of the web application, on a host device (such as). Web Client - In this document, the web client refers to the web platform in which the product/application can be accessed using the internet.

2. General Description

2.1 Product / System Functions

The general functionality of the web application is to provide a unique file hosting/storage system whereby users will be able to access their files from either the desktop client, web client or the mobile phone client on demand. This means that users will be able to accomplish the tasks mentioned above at any time, provided they have internet access. Once the user has successfully signed in using the credentials provided to them by the organisation, they will be met with a home screen consisting of all files uploaded. These files will be viewable as well as removable. Furthermore, the user will have on-screen options to either upload or sync their files manually to the web application. Should they choose to upload, the user can either upload from a particular directory on the host device or simply drag the file/document onto the web application. In conjunction with this, the user can also manually sync their files/documents onto the web application. This can be achieved by pressing a designated button on the web application. Once this button has been utilised, all files added to the desktop client will be automatically synced to the cloud to ensure no data loss.

2.2 User Characteristics and Objectives

The core users of this product will be institutions or organisations with whom privacy is of major importance. Files/data that are stored on public cloud platforms are sent to huge data warehouses that consist of data servers. TheIt will then be organised and stored under your account. The location is unknown to the owner. This raises significant concerns about the issue of privacy. This product is intended for system administrators of these organisations/institutions and also the students/staff.

The system administrators of these organisations or educational institutions will have full control of the files created by students/staff can apply policies that will be beneficial to them (i.e size restrictions, etc.). They will also be able to restore login if, for example, a student has forgotten their login credentials or are having issues uploading a particular file.

For non-experienced users, the system is intended to be simple to use. To create a user-friendly interface. A login/sign up feature will be implemented whereby users will be able to view files/documents related solely to them and not any other users on the system. This will ensure privacy between users. Users will also have the ability to

delete/remove unwanted files, add new files and finally manually sync the files on their host machine (as mentioned previously).

2.3 Operational Scenarios

Use Case Descriptions

User Case 1: User Access - User Register

- Upon entering the web application, the user will be prompted to register or login. If ‘Register’ is chosen by the user, they will be able to register their credentials. If the user is associated with a particular organisation/educational institution(i.e a university) the user will be redirected to the login page of their respective organisation/educational institution.

Use Case 1	**Sign up/Register**	
Goal in Context	User	
Precondition(s):	Application is installed on the user’s host device	
	User has reached the web client’s welcome page	
Success End Condition	Registration is successful and user can access the application	
Failed End Condition	Incorrect credentials, unable to register	
Primary Actor	User	
Trigger	User opens the web/desktop application	
Description	**Step**	**Action**
	1	User has successfully reached the web/desktop client.
	2	User clicks “register”.
	3	User completes the provided form with required information.
	4	User hit “save” button.
	5	User gains access to the application.
Extensions	**Step**	**Branching Action**
	4a	User unable to register as system has detected an error.

User Case 2: User Access - Log in

- Upon gaining access to the web application, if the chooses the option to login, they will be prompted a prompt requesting for the username/email and password in which was used to register. Once the user enters the correct credentials, they will successfully enter the platform and begin work.

User Case 2.1: User Access - Unsuccessful Login attempt.

- If however the user is unsuccessful, an option to reset reset their password. If this option is utilised, then the system administrator of the organisation will be notified. They will then send a reset password link to the account owner via email.

Use Case	**Login**	
Goal in Context	User logs into the system.	
Precondition(s):	User has successfully registered and an account exist on the application.	
Success End Condition	User achieves login.	
Failed End Condition	User does not log onto the system successfully.	
Primary Actor	User	
Trigger	User navigates to the login icon on the user interface.	
Description	**Step**	**Action**
	1	User opens the web/desktop application.
	2	User navigates to the ‘Login’ icon.
	3	User enters their username and password.
	4	User clicks the “Login” button.
Extensions	**Step**	**Branching Action**
	4a	Login is unsuccessful as account/password does not exist.
	4b	User clicks “forgot password”.
	4c	User receives a reset password email.
	4d	User resets their password.
	4e	User retries with new credential.

User Case 3: User Upload file

- Once login has been successful, the user will have the option to upload file. A button will exist on the system whereby if utilised a pop up will emerge revealing all to files located on their host computer and prompting the user to choose a file. Also to provide a convenient and easy-to-use platform, the user will also be allowed to drag the file directly from their desktop to the desktop client of the system.

Use Case	**Upload file/document**	
Goal in Context	Files/documents successfully uploaded onto the web/desktop application.	
Precondition(s):	Login has been accomplished.	
Success End Condition	File/Document is uploaded and stored on private cloud.	
Failed End Condition	File not uploaded due to an error.	
Primary Actor	User .	
Trigger	User clicks “upload file/document” icon.	
	User drags file/document onto the web/desktop client.	
Description	**Step**	**Action**
	1	User successfully logs in.
	2	User wishes to add file.
	2a	User clicks icon to upload file.
	3	Pop up screen emerges with files/documents on user’s host device.
	4	User clicks the required documents.
	5	Documents is uploaded to application.
Extensions	**Step**	**Branching Action**
	2b	User drags the file onto the web/desktop application.
	2b(1)	File/document is uploaded onto the application.
	5a	File does not upload due to an error.
	5a(1)	User resolves the error.
	5a(2)	User tries again.

User Case 4: User Delete/Remove file

- If not satisfied with a particular file uploaded, or if there is insufficient amount of space, users will have the option to delete files. This will move files to a ‘trash’ folder where users will have 30 days to retrieve the item, otherwise the item will be removed indefinitely.

Use Case	**Delete/Remove File**	
Goal in Context	Remove unwanted file	
Precondition(s):	**Unwanted file must exist on the system**	
Success End Condition	Unwanted file is removed.	
Failed End Condition	Unwanted file removal was unsuccessful.	
Primary Actor	User	
Trigger	Dragging the unwanted file into the trash or deleting it using the provided utility.	
Description	**Step**	**Action**
	1	User locates the unwanted file.
	2	User clicks the unwanted file.
	3	A button is highlighted to with “remove” on it.
	4	User clicks ‘remove’.
	5	File is removed.
Extensions	**Step**	**Branching Action**
	5a	File is not removed due to an error.
	5b	User contacts System admin.
	5c	Upon resolution, the removal is re-attempted.

User Case 5: User manually add files

- User will manually sync their files onto the platform. This will be achieved upon gaining access to the web or desktop application. After they have input their credentials and successfully logged in, a button to manually sync will appear on their home screen. Once this button is clicked, a pop up will appear showing all the files on

their host directory. If they find the file to upload, they click on it and this file will be added to the application.

Use Case	**Add File**	
Goal in Context	Add file from host device	
Precondition(s):	Application must be pre-installed	
Success End Condition	File is uploaded to platform	
Failed End Condition	File is not uploaded	
Primary Actor	User	
Trigger	User clicks “Upload file option”	
Description	**Step**	**Action**
	1	User clicks option
	2	Dialog pops up with all host files/documents available for upload
	3	User chooses the desired file
	4	File is uploaded
Extensions	**Step**	**Branching Action**
	4a	File not uploaded due to error
	4a(1)	User fixes the error or contacts admin
	4a(2)	Upon resolution, user re-uploads file

2.4 Constraints

Time.

Due to the proportion and size of the overall product, time will be a major constraint. To accomplish all tasks listed in this functional specification, a huge amount of research must be embarked upon. Good organisational skills must be practised as other assignment deadlines have to be met also.

Resources.

Unfortunately, a huge amount of information about cloud computing relates to products already in the market and not the implementation. As Google becomes more popular, the need for private cloud becomes less. This causes a shortage of accessible information.

Customer Satisfaction

Although surveys were implemented to ensure that customer satisfaction is met we cannot anticipate every use case of every user. This is because technology is constantly evolving.

3. Functional Requirements

Sign Up/Register

- Description - Once the user has successfully gained access to the system, they will have the option to “Sign up” or “Register”. Essentially, this will be presented in the welcoming screen of the User Interface(UI). Upon completion of the registration form, an account will be created and provided to the user with the credentials provided by the User upon registration.
- Criticality - High
- For this component of the overall system, criticality is high. This is because this component enables the creation of distinct accounts per member. This will enable the organisation of users of similar organisations or educational institutions to be organised facilely in the system database.
- Technical issues - To complete a successful login attempt, users must enter the precise information in the field provided. If the user is a member of an organisation or an educational institution (for example, DCU), validation must occur using a level of authentication. If however the incorrect information has been supplied, an error message will be displayed to the user.
- Dependencies with other requirements - This is an independent feature whereby it does not depend on the former or latter components in the system.

Log in

- Description - This component will administer for the registered users. Post-registration of a user on the system, an account will be created on their behalf. To access this account, the user must have chosen a username (in

the form of the email used to register), as well as a password. The user will then employ their username and password in the provided field.
- Criticality - High
- This feature is highly essential to the overall system. Without it, users will be unable to use the platform. To protect a user's information and prevent the breach of privacy, each user must obtain independent accounts. The platform will enable 1 to 1 mapping of accounts, which means, 1 account per registered email address.
- Technical issues - Describes any design or implementation issues involved in satisfying this requirement. The main technical issue in this component would be to ensure that the specified user is validated and linked to the correctly linked to their associated account.
- Dependencies with other requirements - Describes interactions with other requirements. This feature will be fully dependant on the previous component above as it would require each user to register prior their use of the application.

User Profile

- Description - This supplemental feature will be added to the application to provide users with information about them. This will consist of relevant data uniquely associated with the user. This information will be information provided by the User upon registration. The user will have the option to change/edit this information as they wish. However, the user email will not be adjustable as it will confuse the overall project.
- Criticality - (Low)
- The criticality of this feature is low as it's succession or failure does not hinder the functionality of the former or latter components in the overall system. Nevertheless, it is an integral component of the design of the platform.
- Technical issues - CRUD operations will be implemented in the management of the database. However, it is incredibly easy for mistakes to arise from the storage of information to the database. That said, the major technical issue would be to ensure that information is stored correctly (as per user) in the database.
- Dependencies with other requirements - This function is dependant on the registration requirement of the overall system.

Upload Files/Documents

- Description - Once the Web/Desktop client has been accessed, the user will inherit the ability to manually add files to the platform. This is accomplishable in two ways. Firstly, the user will have the capability to drag the file directly (from their host machine) unto the Web/Desktop client respectively. Secondly, the user will be able to complete a similar task by accessing a button on the

platform whereby it enables them to upload the file by manually searching through the documents currently on their host machine. After the user has chosen their required method of file upload, the files/document uploaded will be saved onto the database. For this component, the Apache Cassandra database will be used as it provides security and flexibility.
- Criticality - (High)
- This the criticality for their component is vital. This is because it forms the entire core concept of the project. The omission of this feature will mean that all other components in the platform will be made redundant.
- Technical issues - The technical issues that exists in this component include: Ensuring that the users host files can be accessed by the system, storing the document on the platform where it can be accessed and viewed by the user, guaranteeing that there is sufficient space for the file being uploaded, preventing the deterioration of the quality of the document when being added unto the platform and finally guarantee reasonable upload speeds regardless of the files added.
- Dependencies with other requirements - This component is dependent on component 1, 2 and 4.

The synchronisation of Files/Documents

- Description - There will be two ways in which documents will be uploaded in this system. One way will be manually and the other will be automatically. Manually upload will occur when the user successfully gains access to the web client. Upon this access, the user will have the option to manually sync files from the desktop client which will be pre-installed on the host device. Once this option is utilised, all files on the host device will be synced. The other option is an automatic synchronisation. This occurs on the desktop client-side. Once a user opens the desktop client, any file in which they have will be automatically synced.
- Criticality - (High)
- This is a significant feature in the overall spectrum of the web application. The basis of this application is to prevent loss of data stored on the host computer. Synchronising allows files/documents are uploaded and stored on the cloud platform to prevent unexpected loss of files caused by (for example) network failures on the host machine.
- Technical issues - To accomplish this task, the desktop and web clients must communicate constantly to ensure synchronisation occurs at all times. This can cause a technical issue as it would require multiple calls to the API involved. This can be challenging on one computer alone, however, this web application will be created to allow multiple synchronisations to happen on multiple accounts presumably simultaneously. A resolution for this issue must be found to prevent the application from unexpected stops.
- Dependencies with other requirements - This component is dependant on requirements 1 and 2.

Logout

- Description - This feature will be added to allow the user to log out of the application. This means that the user will be signed out of their respective accounts. Once the allocated button has been tapped, all content that has been synced will remain in their location and stored.
- Criticality - (Low)
- This component provides completeness to the overall system. It is not overly critical but it is essential as it will provide a sense of relief to the user that they have successfully logged out and their data is stored securely.
- Technical issues - Once the user has logged out, they should no longer have direct access to their account until a successful login has been achieved by the user, on the same host device. This will have to be implemented to prevent an unnecessary security breach.
- Dependencies with other requirements - The components that this feature will be dependant is requirement 2. This is because, for the user to log out, they must complete the initial log in.

4. System Architecture

4.1 Architecture Overview Diagram

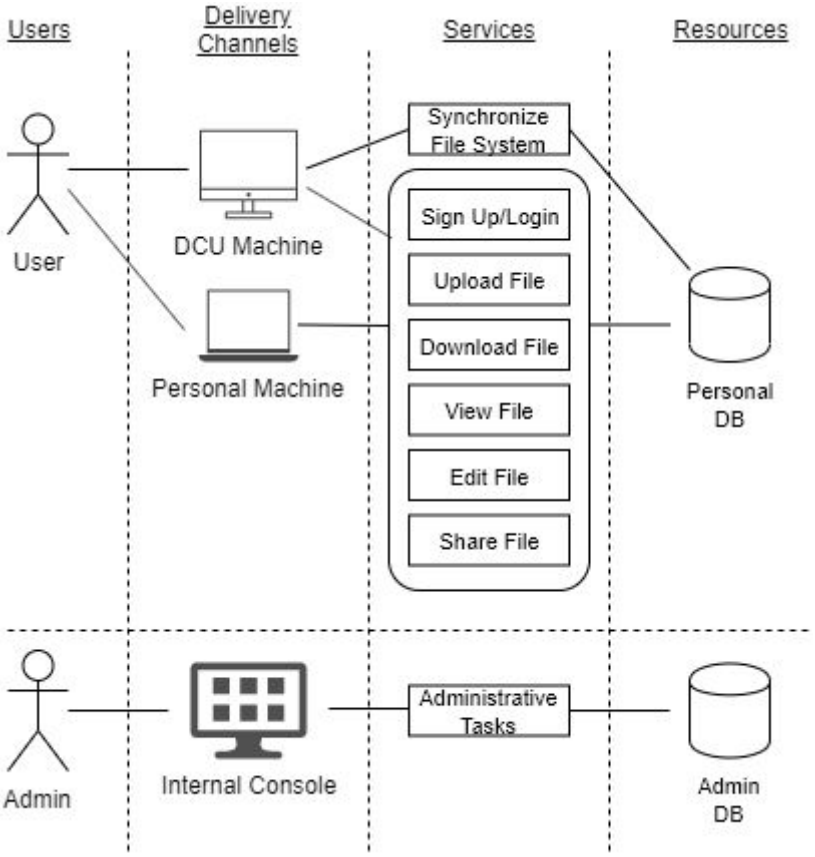


Fig 4.1 above illustrates an Architecture Overview Diagram for the application. The purpose of this diagram is to illustrate, at a conceptual level, an understanding of the system based on its users, delivery channels, the services that it provides, and the resources that it uses.

Users

- A User is anybody who wishes to use the app to store their local data. The diagram above takes the example of DCU as an entity who wishes to use our application, this infers that the user, in this case, maybe a DCU student. The two delivery channels open to them are computers across DCU's campuses, where we assume the desktop application would be installed, and their machines, where they may access the application through the web application.
- An Admin is anybody who is tasked with overseeing administrative tasks for the application. The diagram above presumes that a user and an admin will

be distinct parties, however, this may not always be the case depending on who is the consumer of the application e.g. someone installing the application on their home computer to have their private cloud. The delivery channel available to the admin is the internal console. It is from here that they can access the tools they need.

Services

- Synchronize File System: This service allows a user to automatically synchronize certain directories, saving them from having to manually upload the latest version of a file after every update.
- Sign Up/Login: A user must create an account to use the application as data is separated by user accounts and a user's encryption keys will be tied to their account.
- Upload File: This service allows a user to save files to the application to be accessed from anywhere.
- Download File: This service allows a user to retrieve files from the application from anywhere.
- View File: This service allows a user to view their files and their contents through the web application.
- Edit File: This service allows a user to make changes to their files to be saved in the application.
- Share File: This service allows a user to share their files with another user of their choosing.

Resources

- Personal Database: This is where users file data will be stored along with account information.
- Admin Database: This is where information about the administrator will be stored.

5. High-Level Design

5.1 System Context Diagram

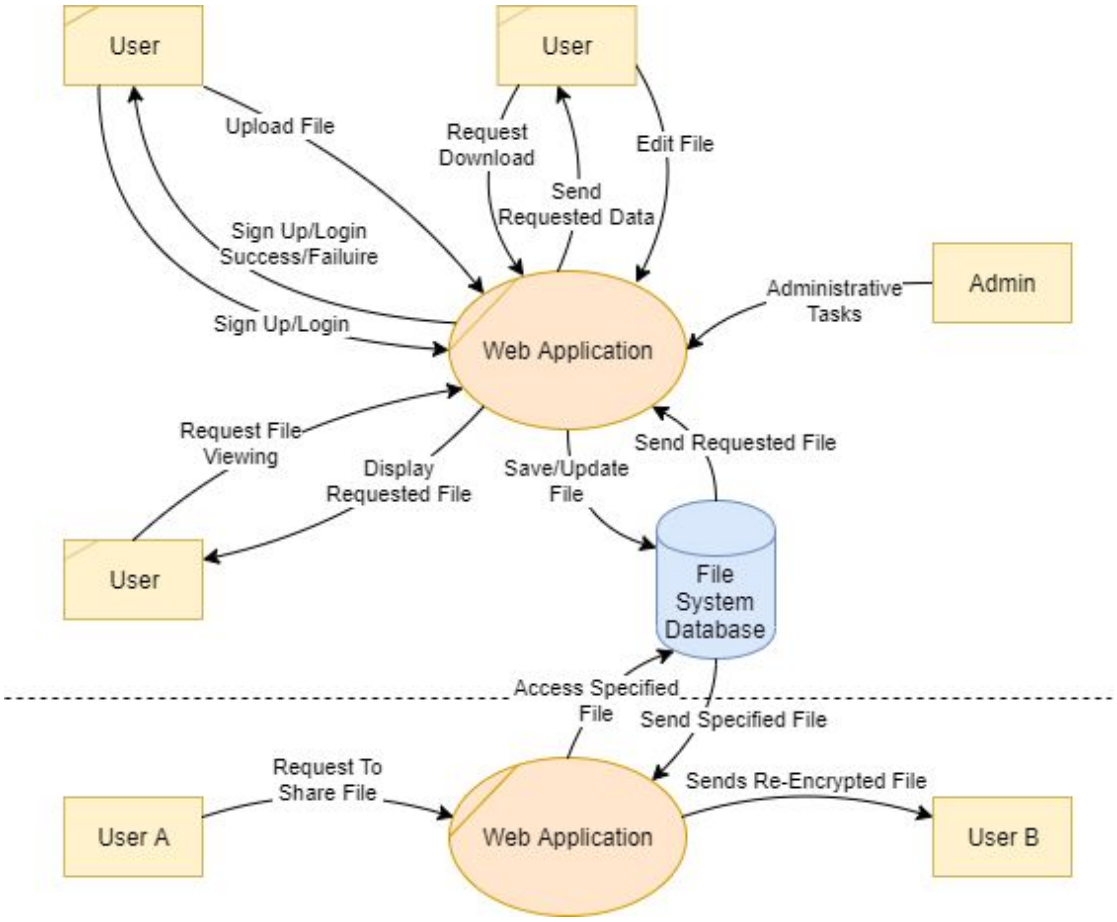


Fig 5.1 above illustrates the System Context Diagram of the application. This diagram defines the boundary between the application and its surrounding environment. It shows the key relationships between the application and external entities along with their inputs and outputs.

As shown in the diagram the User can interact with the application in many ways. Initially, the user must create an account and sign in. From here they can upload and download files, as well as set up automatic synchronization. The user may also view and edit the files that are stored in the application. Any alteration to the stored data that a user requests requires the application to interact with the database to alter the corresponding files. A user may also opt to share some files with another user, this requires the application to request the file from the database and send an encrypted

copy to the other user. To improve the app or maintain fluid usage, the admin may need to issue administrative tasks from time to time.

5.2 Data Flow Diagram

Fig 5.2 above illustrates a simple Data Flow Diagram of the user logging into the application. It presents how the login credentials of the user will be verified to the User Manager and will be checked in the database. Once data has been retrieved from the database, the User Manager will bring back a success/failure login notification.

5.3 Sequence Diagram

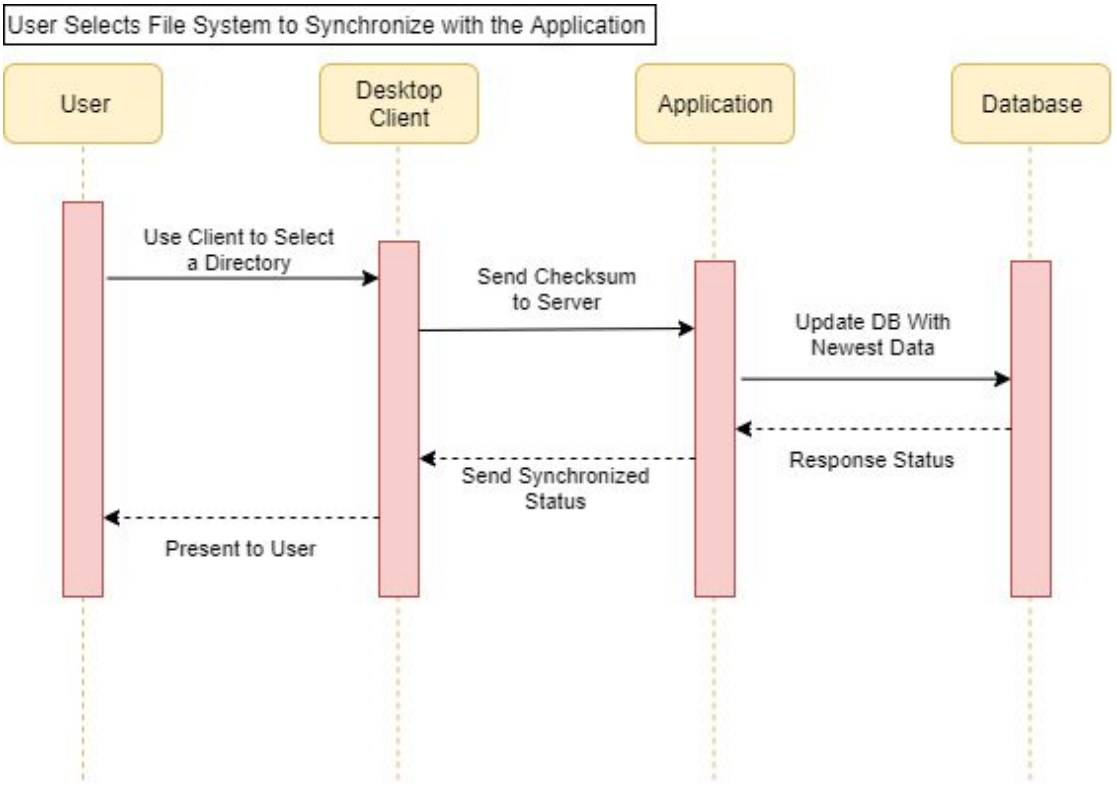
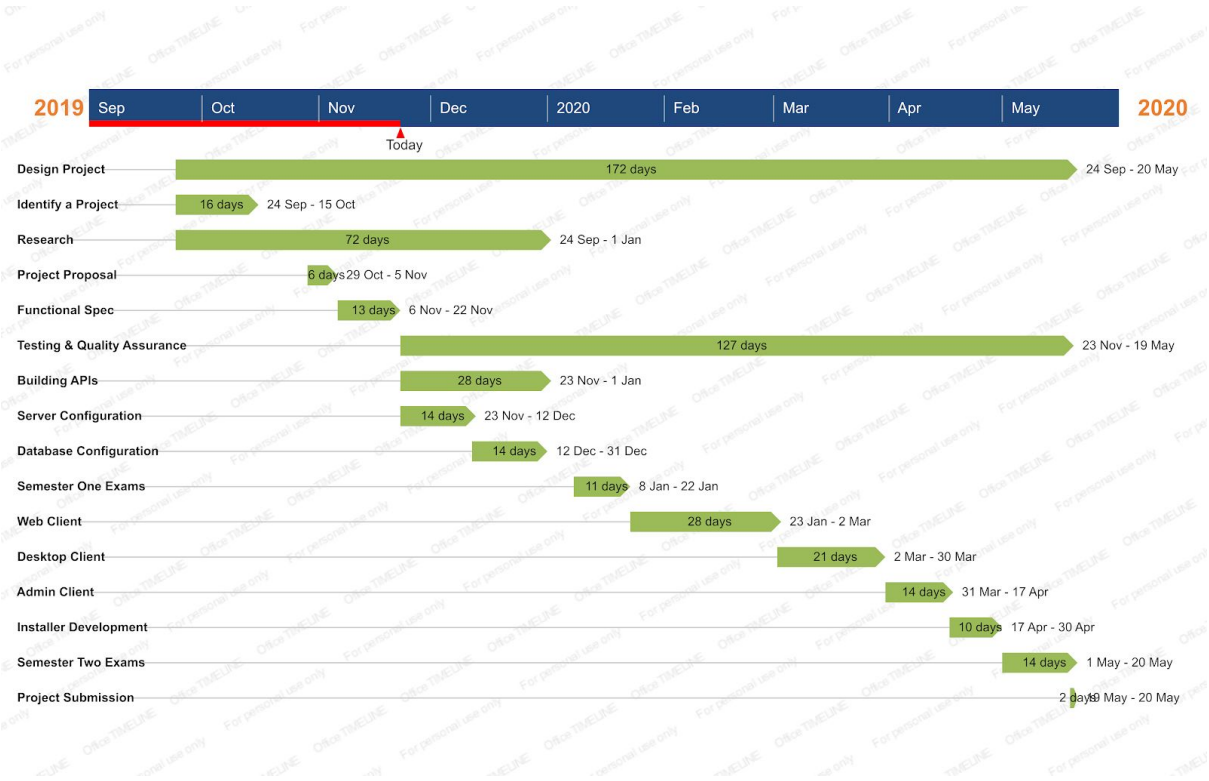


Fig 5.3 above illustrates a Sequence Diagram for when a user wants to set up a directory for automatic synchronization. The user uses the desktop client to select which directory they would like to synchronize. The client then takes a copy of all the files and sends an encoded version of them to the application. From here, the application checks for which version is the newest and updates the database accordingly. The database will return its response status, which will then be relayed through the client and finally to the end-user.

6. Preliminary Schedule



The timeline above was created using Office Timeline and shows the work we have already undertaken and future work that we will take on as well.

7. Appendices

https://cassandra.apache.org/
https://expressjs.com/
https://nodejs.org/en/
https://docs.docker.com/compose/
https://reactjs.org/
https://redux.js.org/
https://facebook.github.io/react-native/
https://electronjs.org/