Pronto

Agile Architects

Group Members:

Name	Description	Email
Andile Ngwenya	Final year BSc Computer Science student at the University of Pretoria. Has some experience in web development and interests in data analytics.	u20612894@tuks.co.za
Jonel Albuquerque	Final year BSc Information and Knowledge Systems student at the University of Pretoria. Joburg West top achiever in information technology with experience in web applications.	u21598267@tuks.co.za
Tyrone Sutherland-MacLeod	Final year BSc Computer Science student at the University of Pretoria with small web development projects done and interests in cyber security.	u21578878@tuks.co.za
Bandisa Masilela	BSc Computer Science student at the University of Pretoria, and former AWS software development engineer intern.	u19028182@tuks.co.za
Shashin Gounden	BSc Computer Science student with experience in web applications and team lead positions in previous projects.	u21458686@tuks.co.za

Last Edit: 2023/05/17

Table of Contents

1.1. Introduction

A. Vision and Objectives

Pronto is a multi-purpose application that will be used by universities, lecturers, and students. Students will be able to create, edit and view custom timetables. Lecturers will be able to schedule reminders for important events like exams and project due dates, as well as notify them of assignment extensions. Institutions will be able to upload their university lecture schedules and campus maps. This application aims to help students organise their schedules and also bring together the features of reminders, navigation, and lecture timetables into a single application.

B. Business Need

Currently, there is no single consolidated application that allows students from any higher education institution to create a timetable based on the modules they are taking. Features for different parts of the application exist in isolation, and we intend to design a system that brings these features into a single place for any student of any university.

C. Project Scope

Defining deliverables

The scope of the project involves developing a proof of concept web/mobile application using Amazon Web Services (AWS) as the underlying infrastructure. The core functionalities and features that we need to develop include:

I. User registration and authentication
- Users, including students, institutions, and lecturers, will be able to create accounts and authenticate themselves to access the application.
- Users will be assigned different roles that will enable them to use the system differently based on their user group.

II. Timetable generation:
- Students will be able to create schedules based on their module choices, and also view assignment deadlines, upcoming tests and examination schedules, and other important events.
- They can also set reminders and define how they would like to receive notifications for upcoming tasks. (E-mail, SMS, or push notifications)

III. Institution pages:
- Institutions will have the ability to create and maintain their own pages within the application.
- They can load and update student schedules, campus maps, assignment deadlines, test and examination schedules, and other relevant information. Students will be able to access these documents and relevant information by visiting their page.

IV. Processing lecture schedules:

The application will allow institutions to upload their lecture schedules in the form of a .csv or .xls file, which will then be processed and stored so that a student can search for a module and have its information generated for their timetable.

V. Notifications:

The application will allow for three notification types that the user can select :

- Push
- Sms
- Email

Students can set which type of notification they would like to receive from these options.

VI. Mobile support

The application should be accessible and responsive on mobile devices, allowing students to conveniently access their tasks and receive notifications on the go.

VII. Web support

Users that are universities or lecturers will use an application with a web view to create their accounts, manage their pages, and set reminders respectively.

VIII. Reminders

The application will provide reminders to students for upcoming deadlines, events, or any updates from their institutions. These will be presented to the user via the notifications subsystem

Additionally, the optional functionalities and features that we would like to develop include:

IX. Geo-location services

Students should be able to use a map service within the application that will give them directions to a lecture hall from their current location. This service will use a walking distance calculator to work out the time required to navigate between the lecture venues.

X. Extracting and transforming the lecture schedule

When an institution uploads its lecture schedule, we would like to take the provided format and transform it into a standard format for storage and consistency purposes. With this, inconsistencies with naming formats can be addressed.

As it stands, we plan to be finished with the core deliverables by the 24th of July 2023. Thereafter, development on the optional deliverables will begin and finish by the 20th of September 2023.

D. Identifying constraints

- All modules and tools used outside of AWS services should be open-source and documented.
- All data that we store needs to be stored and used securely. The privacy of users should be taken into account with the design, making sure to consider POP, GDPR, PII, etc.

*[insert milestones here once decided]

1.2. User Stories/Characteristics

A. Students

1. As a student, I want to be able to create a custom timetable based on the lectures I have, so that I can stay organised and have an easy way to make a timetable.
2. As a student, I want to be able to view assignment deadlines, upcoming tests and examination schedules, and other important events, so that I can plan my schedule accordingly.
3. As a student, I would like to define how I would like to receive reminders for my upcoming tasks so that I can receive notifications in a way that works best for me.
4. As a student, I want to be able to receive notifications about upcoming deadlines or events, so that I never miss important information.
5. As a student, I want to be able to access my institution's page, so that I can easily find relevant information and documents.
6. As a student, I want to be able to use a map service within the application that will give me directions to a lecture hall from my current location, so that I can easily navigate around campus.

B. Lecturers

1. As a lecturer, I want to be able to schedule reminders for important events like exams and assignment due dates so that my students are aware of upcoming deadlines.

2. As a lecturer, I want to be able to change assignment deadlines or test dates as the need arises, and for my students to be notified of these changes.

3. As a lecturer, I want to be able to send out important announcements regarding tests or assignments to my students so that they can be notified immediately of important changes.

C. Institution

1. As an institution, I want to be able to upload files like our university lecture schedules, campus maps, and other relevant information so that students can easily access it.

2. As an institution, I want to be able to update lecture schedules, and other relevant information about the institution, so that students always have access to the most up-to-date information.

D. Any user

1. As a user, I want all of my data stored safely and for the privacy of my data to be taken into account so that my personal information is protected

1.3. Functional Requirements

Requirement 1: User Registration and Authentication

Sub-Requirement 1.1

Pronto will allow users to create a new account by providing their name, email, and password. Accounts will be divided into student accounts and lecturer accounts.

Sub-Requirement 1.2

Pronto shall provide the capability for users to log in to the application using their email and password. Users can log in as either students or lecturers with different privileges.

Sub-Requirement 1.3

Pronto will allow students to pick which university they attend after signing up to get content related to their university.

Sub-Requirement 1.4

Pronto will allow all users to edit their account details, like their name, phone number, and password.

Requirement 2: Timetable Generation

Sub-Requirement 2.1

Pronto shall provide a search capability for modules to students.

Sub-Requirement 2.2

Pronto will allow students to delete modules from their subject list.

Sub-Requirement 2.3

Pronto will allow students to add modules to their subject list.

Sub-Requirement 2.4

Pronto will allow students to edit their timetable by selecting activities for each module.

Requirement 3: File Upload/Retrieval

Sub-Requirement 3.1

Institutions can update student schedules, campus maps, assignment deadlines, test and examination schedules, and other relevant information.

Sub-Requirement 3.2

Students can access these documents and relevant information by visiting their institution's page.

Sub-Requirement 3.3

Institutions can upload their lecture schedules in the form of a .csv or .xls file.

Requirement 4: Notifications

Sub-Requirement 4.1

The application will allow students to select from three notification types (push, SMS, email).

Sub-Requirement 4.2

The students will be able to receive reminders for upcoming deadlines, events, or any updates from their lecturers.

Requirement 5: Reminders

Sub-Requirement 5.1

Lecturers can set reminders for due dates, test dates and any important events that will be sent to students.

Requirement 6: Geo-Location Services (Optional “wow factor” for later)

Sub-Requirement 6.1

Students can use a map service within the application that will give them directions to a lecture hall from their current location.

Sub-Requirement 6.2

Students should be able to see the time required to navigate between the lecture venues.

Sub-Requirement 6.3

Pronto will allow lecturers to interactively pin locations of lecture venues.

1.4. Quality Requirements

1. Performance Requirements

A. Availability

Definition

The tool should always be available for all users and any unavailability communicated.

Acceptance Criteria

- The website and application should have an uptime of **90%**
- Any planned maintenance will occur outside of normal university hours
- Planned downtime will be indicated or communicated with clear UI changes.

B. Scalability

Definition

The product should be at first able to scale horizontally to accommodate for typical traffic that would be expected by an institution during peak times, then this should extend out to multiple institutions per region.

Acceptance Criteria

- Autoscaling of the microservices used will be able to accommodate changing traffic on a regular basis.
- Different instances of the back-end will be deployed per major region to decouple scaling, allowing for different regions and resources to scale up accordingly.

C. Performance

Definition

The different clients of the product should be fast and highly responsive.

Acceptance Criteria

End-to-End testing will be conducted regularly to test server response times and ensure that a user can quickly perform each use case or multiple use cases in an acceptable time frame.

2. Usability

Definition

Users should be able to navigate and interact with the main features of the tool with ease and intuitively.

Acceptance Criteria

- We will conduct regular tests with each UI implementation and ask our user to rate their experience
- The reviews will be conducted anonymously using a google form for data collection and rating
- Separate aspects will be scored accordingly
- The user experience should have a **80% satisfactory rate**

3. Security

Definition

User data should be secured at rest and in transition in order to prevent unauthorised access and comply with local laws and regulations.

Acceptance Criteria

- User data will be Encrypted in transition and Encrypted at rest.
- Users will be grouped into user pools during sign up
- This will provide limited access to data modification and access as per use case.

4. Maintainability

Definition

The software should be easily maintainable by both internal, or new external party(s), with clear concise documentation and code.

Acceptance Criteria

- Code linters and standard linting rules will be applied to ensure code uniformity.
- A 3-person review procedure will be used to ensure that all solutions provided are efficient and employ the best procedures.
- New features implemented will be explained thoroughly with each **Pull Request**, which will also be tested and evaluated by at least 3 parties prior to deployment.
- The application will be a serverless in it's core, which means that other than optimising implementation, there won't be a server to maintain.

5. Reliability

Definition

The software should be stable and present reliable and verified data, with low downtime and errors.

Acceptance Criteria

- Lectures will be asked to confirm their scheduled events prior to upload
- Students will be asked to confirm their courses after selection
- Error logging will be performed so that its nature can be assessed and prevented in the future.

1.5. Design patterns

1.6. Overview

The system will be using a microservices architecture where each independently deployable service will focus on executing at least one business requirement according to the Functional Requirements.

Service	Provider	Role
Amazon Amplify	Amazon Web Services	It will be used to build, deploy and host both the web and mobile application. It will also be used to instantiate and deploy new resources as deemed fit.
Amazon Cognito	Amazon Web Services	It will be used for sign-up,authorization, authentication and user grouping.
Amazon Simple Storage Service	Amazon Web Services	It will be the blob storage database that will store any files uploaded by the institution admin such as campus maps and academic year schedules.
Amazon DynamoDB	Amazon Web Services	A noSQL database that will be used to store courses, activities and any future scheduled activities and/or reminders.
Amazon AppSync	Amazon Web Services	Used to build a graphQL API, parse and resolve all graphQL queries/Subscriptions and Mutations. Connect users to other deployed services such as enabling admins to upload files directly to S3.
Amazon Simple Notification Service	Amazon Web Services	Used to send push notifications to the application and other platforms such as emails, according to the user's preferences with push notifications being the default.
Amazon Event Bridge	Amazon Web Services	It will be used to schedule and trigger notifications for new activities created by lectures. The notifications will be sent using Amazon SNS,
Amazon Glue	Amazon Web Services	Used to extract the needed data from the academic schedule uploaded by the institution. It will transform the data to the expected standardised schema and load it onto the course information on our Course Info database (dynamoDB).
Amazon Lambda	Amazon Web Services	It will be used as an API Handler for our graphQL API and datastore resources. Multiple Cloud functions for executing our requirements will be built and hosted on AWS Lambda.

1.7. Service Contracts

Pronto makes use of GraphQL.Thus the front end uses queries and mutations to communicate with the backend. The format of these
is described below

```
const resp =await API.graphql( { queries: myQuery,
                                 variables: { input : { id :”XXX”,name:”John”}},
                                 authMode:"AMAZON_COGNITO_USER_POOLS" })
```

Below are the different queries/mutations ,their inputs and the structure of the JSON objects they return

Queries

```
query getLecturer{
        getLecturer(id:String){
          id
          name
          firstname
          email
      }

{
        "data" : {
            "getLecturer" : {
                    "id":String,
                    "name":String,
                    "firstname":String,
                    "email":String
                    }
            }
}

query getStudent{
        getStudent(id:String){
          id
          name
          firstname
          email
}

{
        "data" : {
            "getStudent" : {
                    "id":String,
```

```
                        "name":String,
                        "firstname":String,
                        "email":String
                        }
                }
}

query getAdmin{
        getAdmin(id:String){
          id
          name
          firstname
          email
      }


{
        "data" : {
              "getAdmin" : {
                        "id":String,
                        "name":String,
                        "firstname":String,
                        "email":String
                        }
                }
}


query getInstitution{
                getInstitution(id :String){
                id
                adminId
                name
                campusMapUrl
                openingTime
                closingTime
                minimumDuration
              }
}

{
        "data" : {
                  "getInstitution" : {
                                "id" :String,
                                "adminId" :String,
                                "name" :String,
                                "campusMapUrl" :String,
                                "openingTime" :String,
```

```
                                    "closingTime" :String,
                                    "minimumDuration" :number
                                  }
                }
        }


query getCourse{
                getCourse(id :String){
                  id
                  name
                  code
                }
        }

{
          "data" : {
                  "getCourse" : {
                                  "id" : String,
                                  "name" : String,
                                  "code":String
                                }
                }
        }



query getActivity{
              getActivity(id :String ) {
                  id
                  name
                  group
                  day
                  start
                  end
                  venue
                  frequency
              }
        }

{

              "data" : {
                      "getActivity" : {
                                      "id" : String,
                                      "name" :String,
                                      "group" :String,
                                      "day" :String,
```

```
                    "start": String,
                "end" :String,
                "venue" :String,
                "frequency" :number
            }
        }
    }
```

Mutations

```
mutation createAdmin{
    createAdmin( input : {
                    firstname:String,
                    lastname:String,
                    email:String,
                    userRole:String
                }
            )
     {
      id
      }

}

{
    "data" : {
        "createAdmin": {
                "id":String
            }
        }
    }


mutation createStudent{
     createStudent( input : {
                    firstname:String,
                    lastname:String,
                    email:String,
                    userRole:String
                }
            )
     {
      id
      }

}
```

```
{
        "data" : {
                "createStudent": {
                        "id":String
                    }
            }
}


mutation createLecturer{
         createLecturer( input : {
                        firstname:String,
                        lastname:String,
                        email:String,
                        userRole:String
                    }
              )
          {
           id
           }

}


{
        "data" : {
                "createLecturer": {
                        "id":String
                    }
            }
      }


mutation createInstitution{
        createInstitution( input : {
                        name: String,
                        adminId:String,
                        campusMap:String,
                        openingTime:String,
                        closingTime:String,
                        minimumDuration:String,
                        domains:[String]
                       }
                 )
          {
            id
            adminId
```

```
            }
}

{
            "data" : {
                    "createInstitution ": {
                                        "id" :String,
                                        "adminId" : String,
                                       }
                    }
}



mutation createCourse{
            createCourse( input : {
                                name : String,
                                institutionId: String,
                                code:String
                                   }
                        )
            {
              id
              institutionId
            }
}

{
            "data": {
                    "createCourse" : {
                                        "id" : String,
                                        "institutionId" : String
                                      }
                   }
}

mutation createAnnouncement{
             createAnnouncemnent( input : {
                                        courseId: String,
                                        description: String,
                                        date: String
                                      }
                                )
            {
              id
              courseId
              createdAt
            }
```

```
}


{
            "data": {
                    "createCourse" : {
                                        "id" : String,
                                        "institutionId" : String,
                                        "createdAt": String
                                       }
                   }
          }

mutation createActivity{
                createActivity(  input: {
                                        name:String,
                                        courseId:String,
                                        day:String,
                                        start:String,
                                        end:String,
                                        group:String,
                                        frequency:number
                                       }
                                )
                 {
                   id
                   courseId
                 }
}

{
            "data" : {
                      "id":String,
                       "courseId" :String
                     }
}


mutation updateAdmin{
            updateAdmin( input : {
                                    id:String,
                                    institutionId,
                                   }
                         )
             {
               institution {
                    adminId
                    closingTime
```

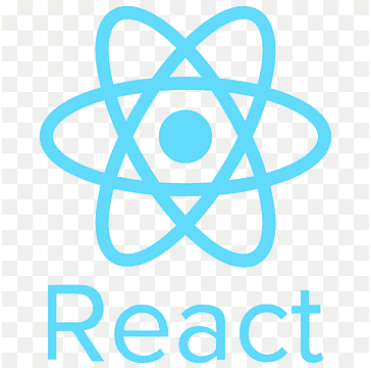
```
                name
                campusMapUrl
                createdAt
                minimumDuration
                location
               }
          }
}

{

          "data" : {
                "updateAdmin": {
                        "institution":{
                                                "adminId":String,
                                                "closingTime" :String,
                                                "name" : String,
                                                "campusMapUrl":String,
                                                "createdAt" :String,
                                                "minimumDuration" : String,
                                                "location" :String
                                                        }
                        }
                }
}
```

2.2. Tech Requirements

1. Presentation/View: The front end will be built using the React framework. For mobile viewports, React Native will be used and for the web view ReactJS will be used.



2. Authentication: Pronto shall utilize Amazon Cognito for user authentication and authorization. This service will handle user sign-up, sign-in, and management of user credentials, ensuring secure access to the application's functionalities.

3. Service: Pronto shall implement a GraphQL API using AWS AppSync or AWS API Gateway. GraphQL will enable efficient data retrieval and manipulation by providing a flexible and precise query language. This API will serve as the primary communication interface between the frontend and backend components of the application.

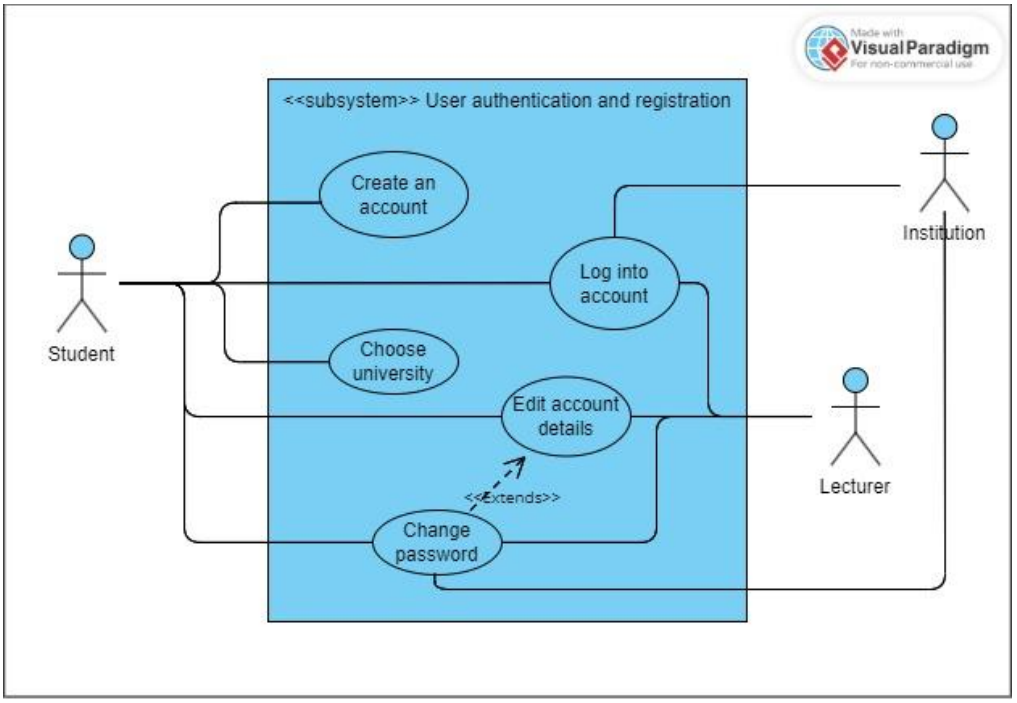
4. Database: Pronto shall employ Amazon DynamoDB as the backend database for storing and retrieving scheduling data. DynamoDB's scalability and performance will enable efficient management of large volumes of scheduling information. The database schema shall be designed to effectively represent the entities and their relationships.

5. Blob Storage: The scheduling application shall utilize Amazon S3 for storing and serving files such as campus maps, logos, or any other blob-like data associated with the scheduling system. S3's durability and scalability will ensure reliable storage and retrieval of files while providing a secure access mechanism.

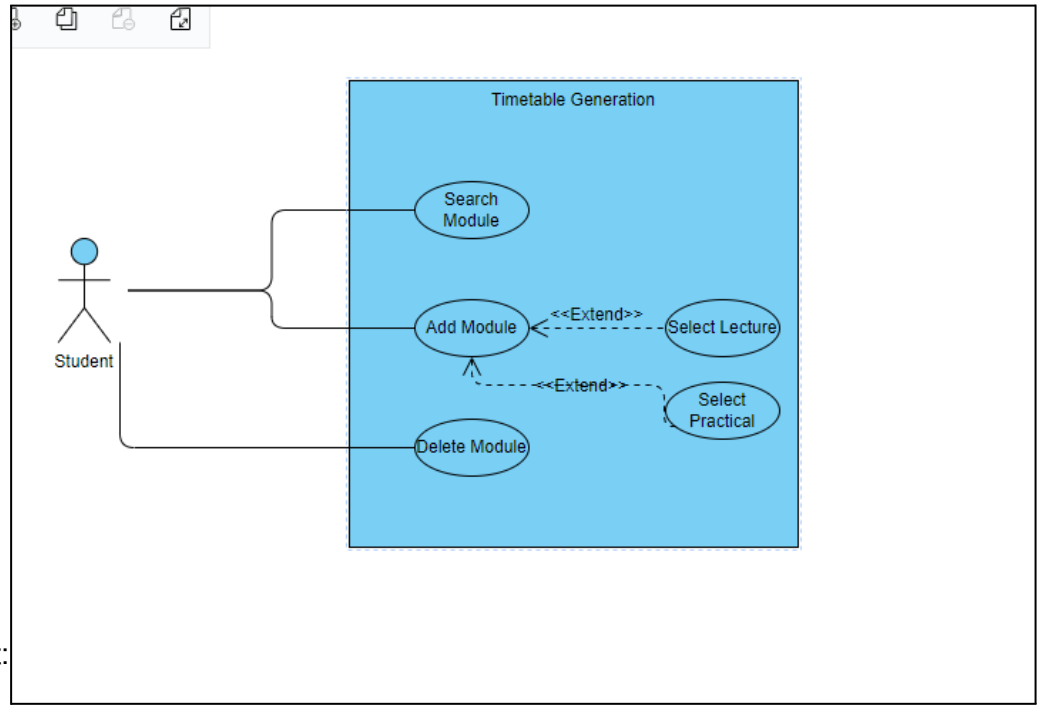
2.3. Diagrams

A. Use case Diagrams

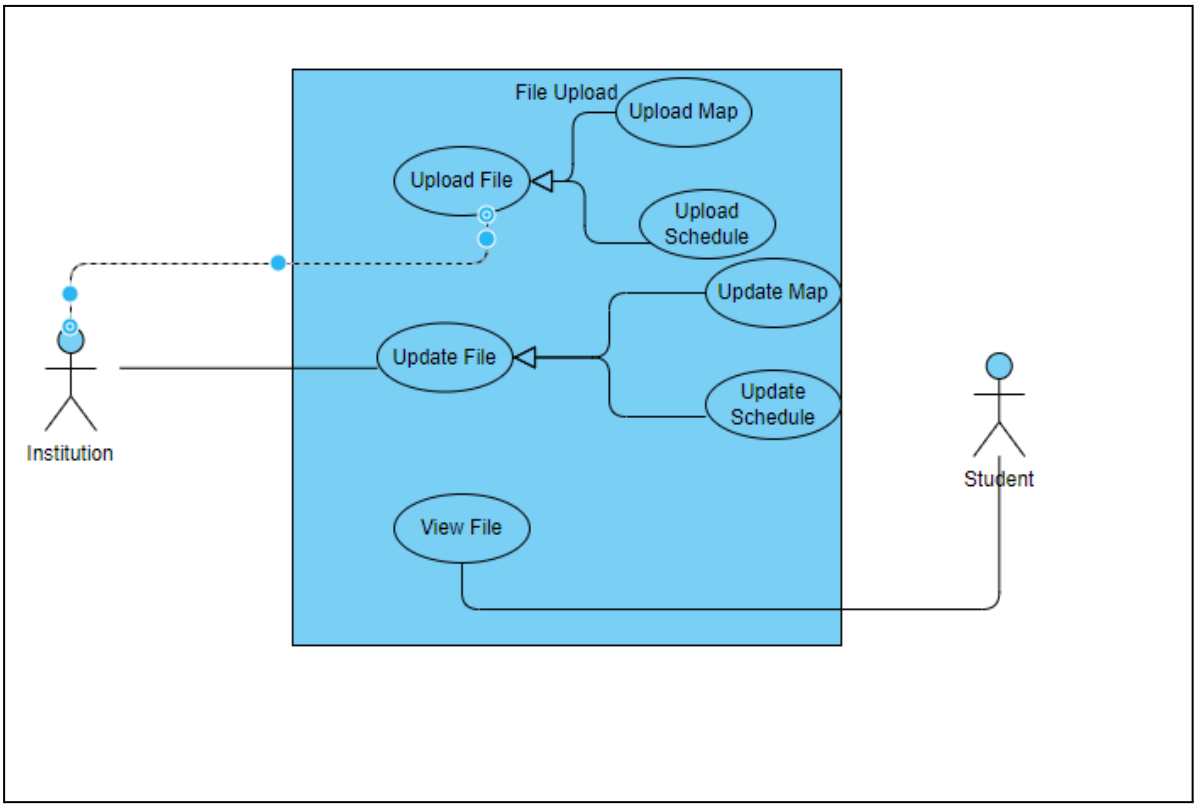
1. User Registration and Authentication



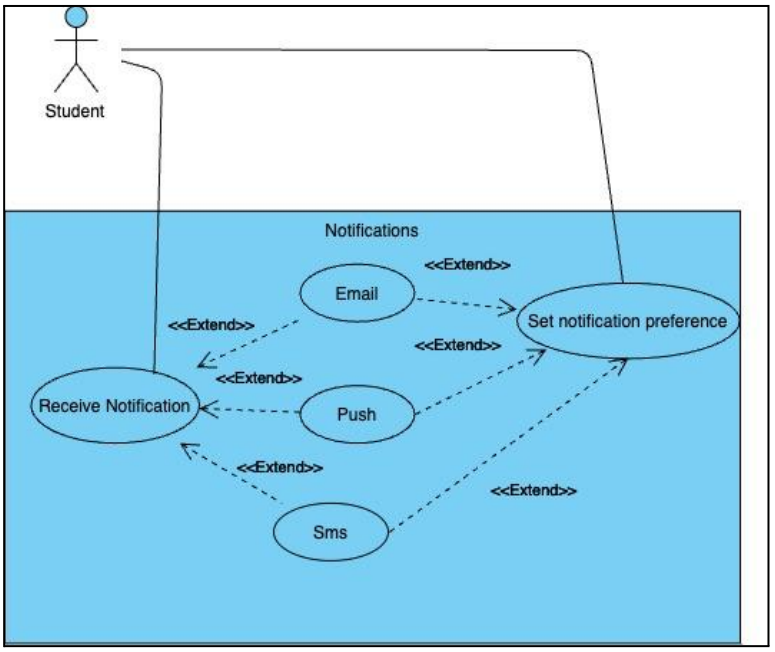
2. Timetable Generation



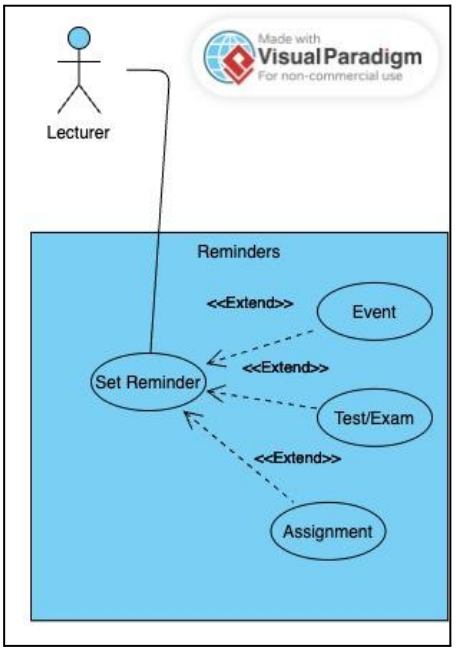
3. File Upload/Retrieval



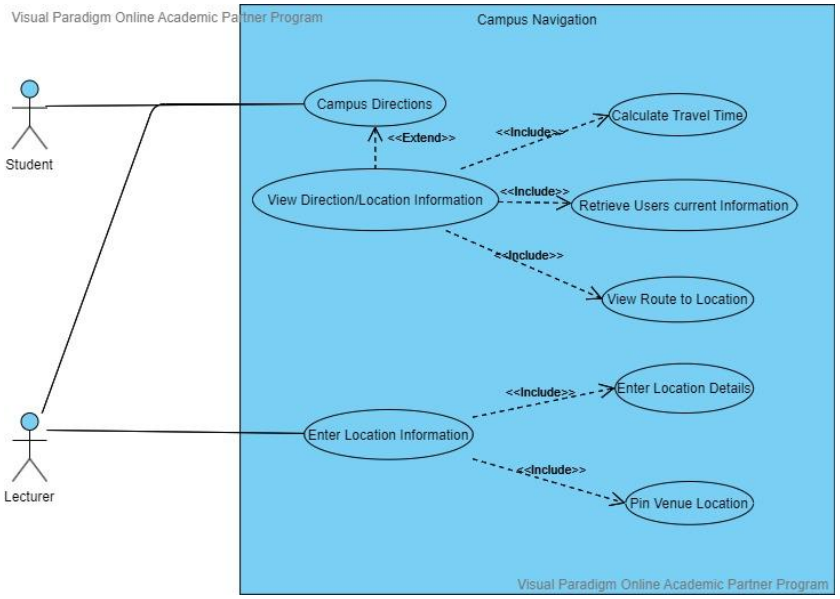
4. Notifications

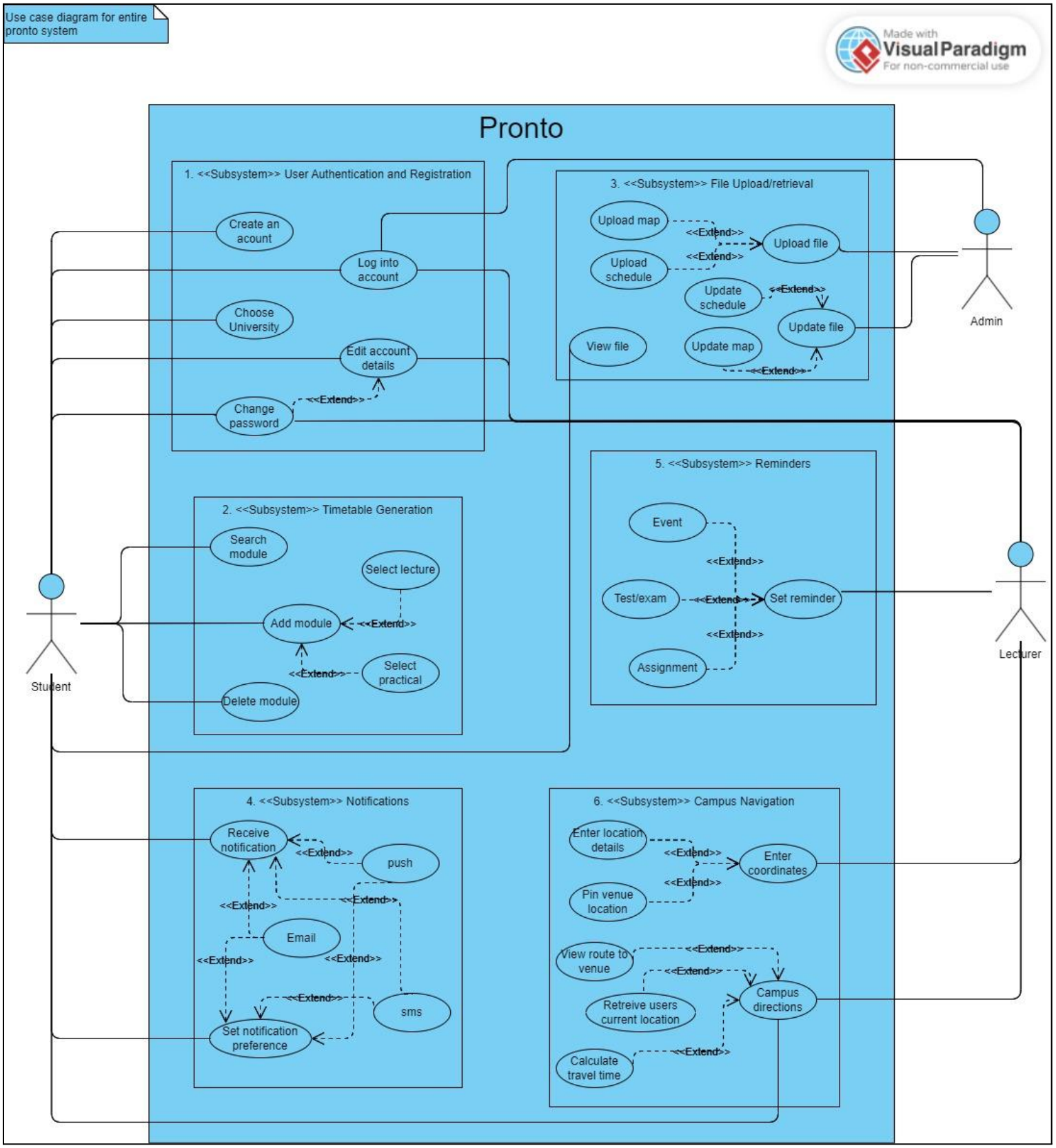


5. Reminders

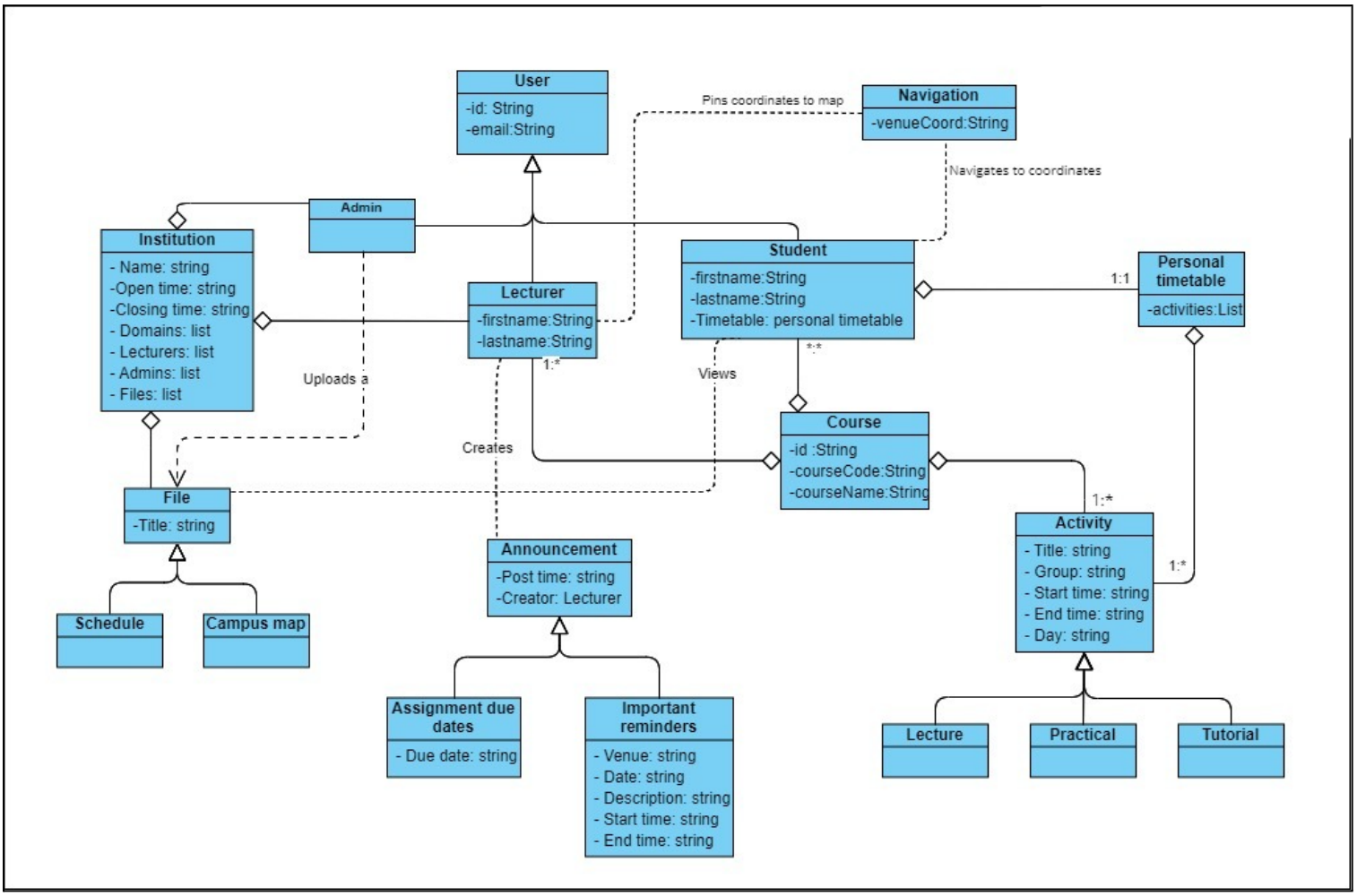


6. Campus Navigation




Use case diagram for entire pronto system
Made with Visual Paradigm For non-commercial use
Pronto
1. <<Subsystem>> User Authentication and Registration
Create an acount
Log into account
Choose University
Edit account details
Change password
<<Extend>>
3. <<Subsystem>> File Upload/retrieval
Upload map
Upload schedule
Upload file
Update schedule
Update file
View file
Update map
<<Extend>>
2. <<Subsystem>> Timetable Generation
Search module
Select lecture
Add module
Select practical
Delete module
5. <<Subsystem>> Reminders
Event
Test/exam
Set reminder
Assignment
4. <<Subsystem>> Notifications
Receive notification
push
Email
sms
Set notification preference
6. <<Subsystem>> Campus Navigation
Enter location details
Enter coordinates
Pin venue location
View route to venue
Retreive users current location
Campus directions
Calculate travel time
Admin
Student
Lecturer

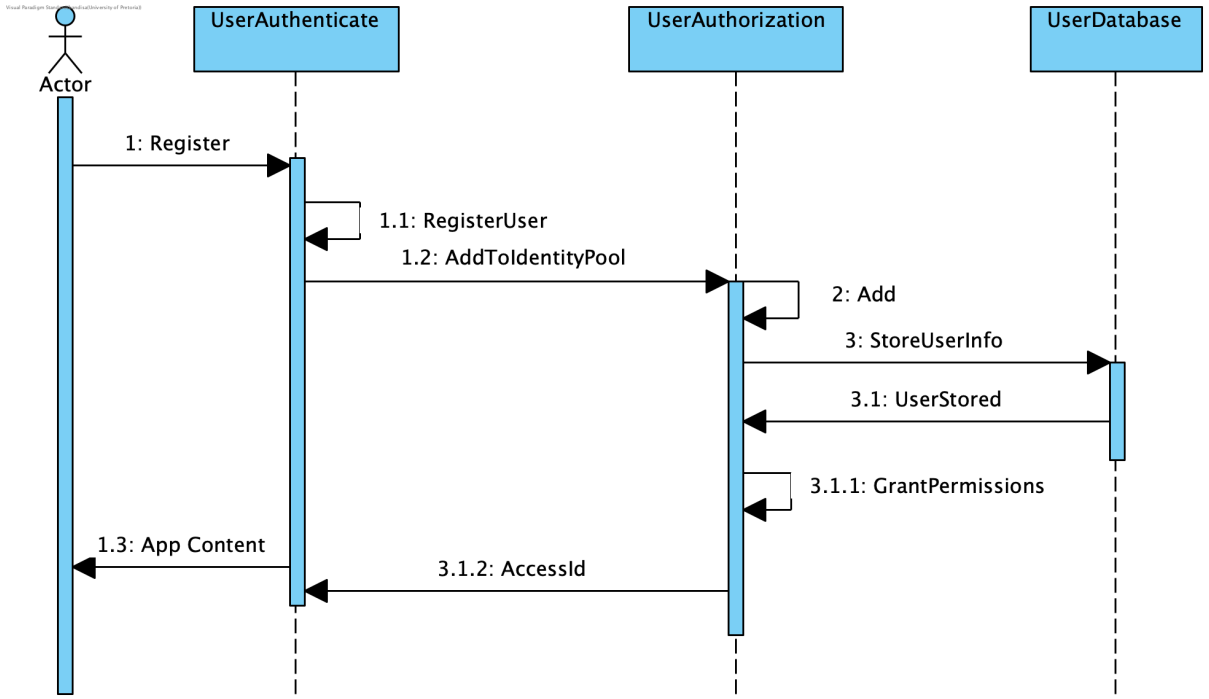
B. Class Diagram



C. Sequence Diagrams

1. Authentication and Authorization

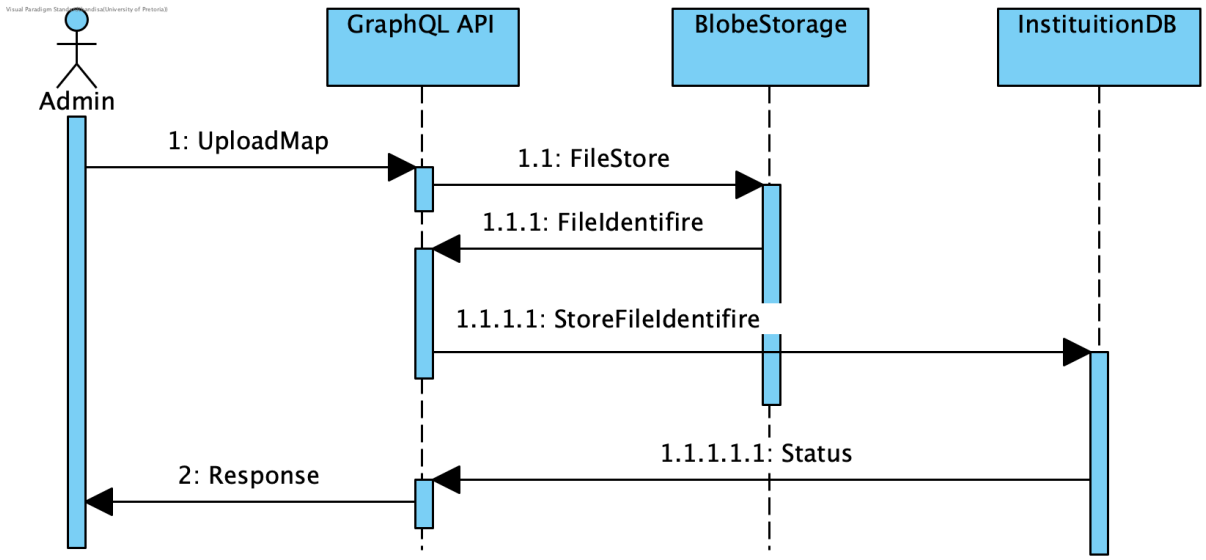
Sequence Diagram



2. File Upload

A. Campus Map

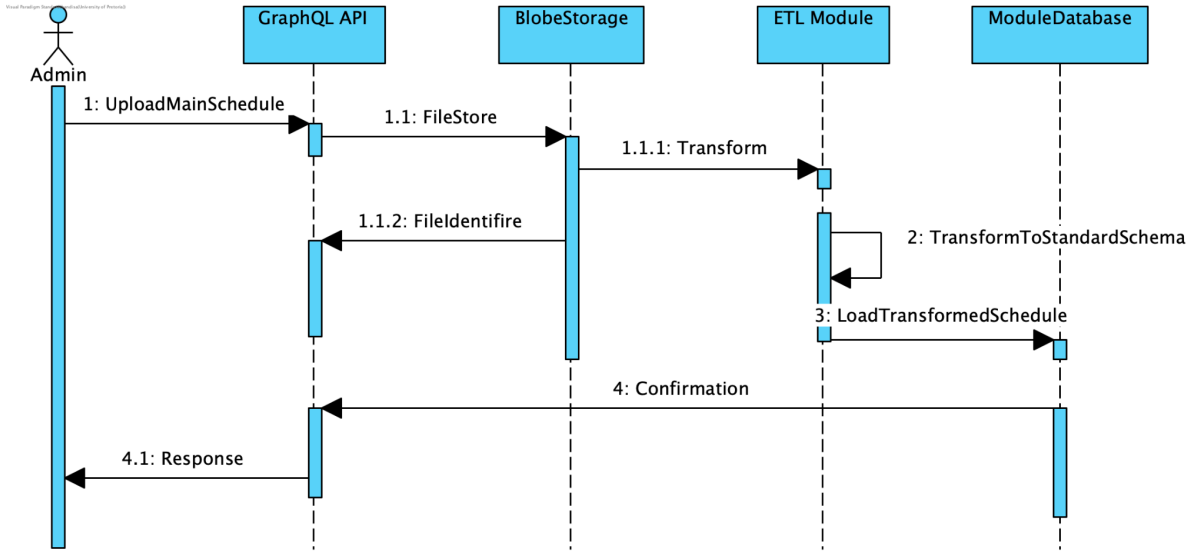
Sequence Diagram



Last Edit: 2023/05/17

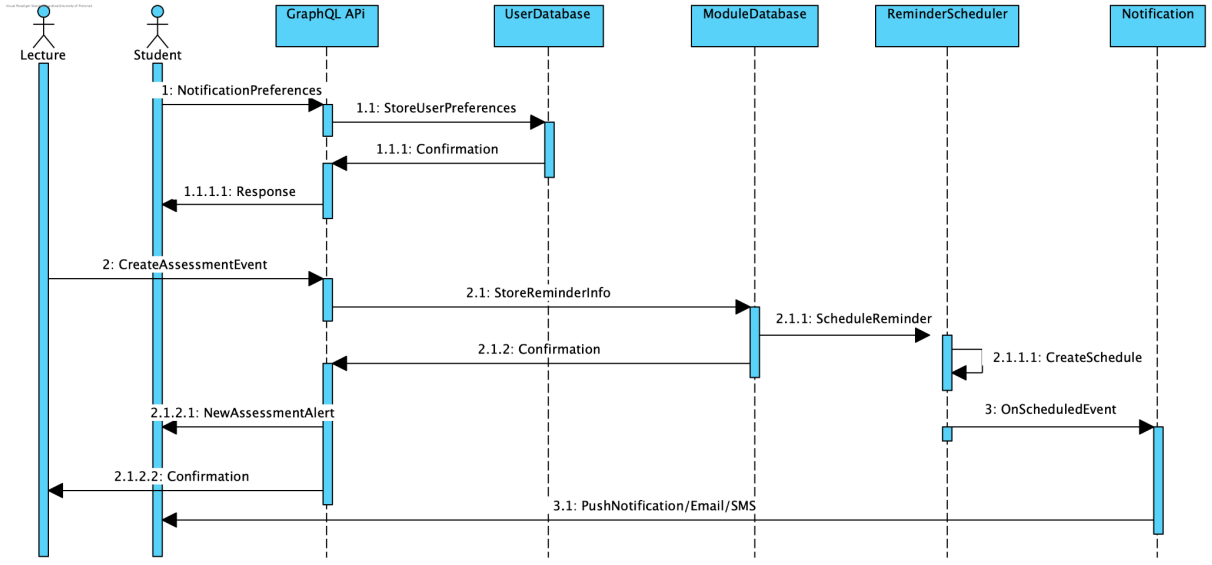
B. Transform Schedule to Standard Schema

Sequence Diagram



3. Notifications and Reminders

Sequence Diagram



D. High-Level Architecture

High-level architecture of the entire system with the microservices used