|  |
| --- |
| **Assignment Cover Letter**    **(Group Work)** |

****

**Student Information**: **Surname**  **Given Names Student ID Number**

1. **Tandra** **Vincentius** **2301894804**

2. **Tanjaya** **Ravel** **2301890320**

3. **Yowen Yowen 2301902390**

4. **Santoso William 2301890390**

**Course Code**  **:ISYS6169**  **Course Name**  **: Database Systems**

**Class** **: L3AC** **Name of Lecturer(s)** **:** **Nunung Nurul Qomariyah**

|  |
| --- |
|  |

**Major** **: CS**

**Title of Assignment** :

NoseBee

**Type of Assignment**  **: Final Project**

**Submission Pattern**

**Due Date**  **:** Week 13  **Submission Date**  **:** Week 13

Problem Description:

The problem that we plan on tackling with this project is an efficient learning management system to facilitate the needs of students, teachers and staff alike. Due to the COVID-19 pandemic, a lot of services are going online including schools. Nowadays, most, if not all schools have resorted to online learning as a way to deliver materials to their students. Whether it is through ZOOM, Google meets or Microsoft Teams, every school has one thing in common now that we are stepping into the age of technology, and that is the need for an efficient online learning management system.

So what defines an efficient learning management system? Well firstly it has to be fast. To ensure efficient learning for the students, learning management systems are required to be fast. Secondly, we need to ensure the system is available at all times. This is to ensure it is able to provide service to all users (students, teachers, staff) at any time of the day 24/7. This is especially important to keep track of schedules for the students.

Not limited to storing student and lecturer schedules, our database is also going to store information related to both the students and lecturers alike. The database is going to keep track of the basic details of each student and lecturers such as their name and gender, the majors present in the school, all the different classes and courses, the scores of each student, the student’s absences as well as their login credentials to enter the system.

Prior to the project proposal, we were instructed to perform an interview with our target users to gather several insights regarding our project. Our group was recommended to interview Ms. Desy from lecturer services for several inputs regarding the current situation in BinusMaya. From this interview, she pointed out that BinusMaya has several problems regarding navigation. There were just too many drop down buttons and it would get confusing for newcomers as they struggle to find the right menu. When she was asked regarding the infrastructure of the database, she said she did not know much about the technicalities so we were unable to gather some pointers for database design.

After further consideration, we realized the navigation problem on BinusMaya was more of a front-end problem and did not have much to do with the database design and unfortunately, we were not able to interview another interviewee. However, although we were unable to get insights on how to design a database system for an efficient LMS, we realized that we as students are technically also valid interviewees since we are one of the target user (students) and thus will be using the application as well and so the design and processes of this project was done by our judgement of what an efficient learning management system is as students.

Teams:

Ravel:

I was responsible for creating the server (written in Go, it also direct interface to our mysql DB and also acts as security layer to prevent unauthorized access to the database) and the frontend (written in Flutter Web).

Vincentius:

I was responsible for helping create some of the queries and the testing of the application. I also held an interview for better realization of our problem description and final solution.

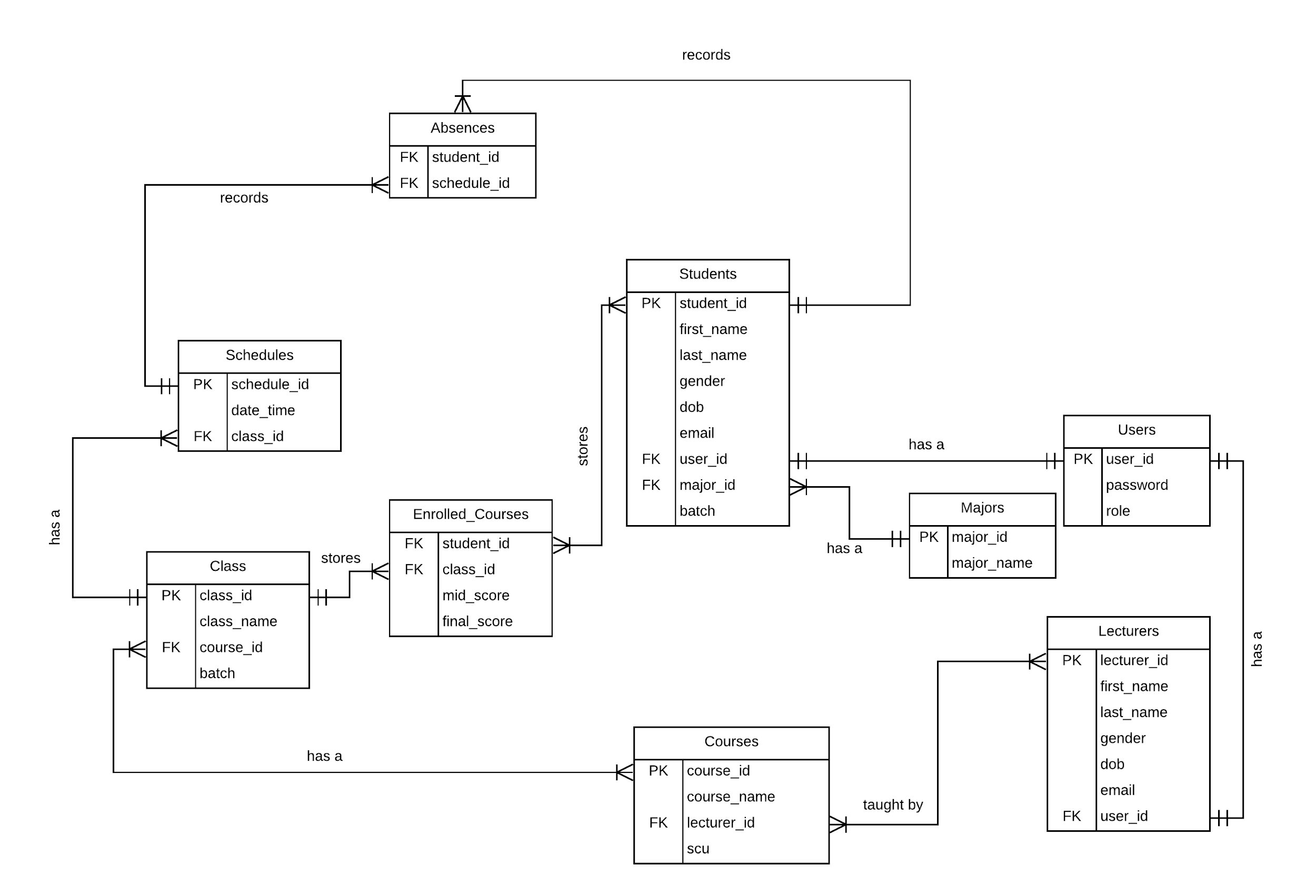
William:

I was responsible for helping a little bit in the queries, contributing to some parts of the report and recording the application demo video.

Yowen:

I was responsible for creating some of the queries, contributing to several parts of the report and analyzing the normalization of our tables.

Entity-Relationship (ER):



Relations:

Users have 3 attributes:

1. user\_id is a primary key with varchar(60) as a data type and not null. The attribute is going to be used in both the Students & Lecturers relations.This attribute will be used to store the user id that the student, teacher, and the staff are going to use to log in to their account.
2. password has varchar(60) as a data type and not null. This attribute will be used to store the student & the teacher’s password which they are going to be used to log in along with their user id.
3. role has char(1) as a data type and not null. This attribute will store the user’s roles from student, teacher, and staff.

Major have 2 attributes:

1. major\_id is a primary key with varchar(60) as a data type and not null. This attribute will store the id of a major that will be available in the college.
2. major\_name has varchar(40) as a data type and not null. This attribute will store the major’s name that is available in the college.

Students have 9 attributes:

1. student\_id is a primary key with varchar(60) as a data type and not null. The student\_id are going to be used in Absences & Enrolled\_Course relations. This attribute will be used to store the id of the student.
2. first\_name has varchar(40) as a data type and not null. This attribute will be used to store the student's first name.
3. last\_name has varchar(40) as a data type and not null. This attribute will be used to store the student's last name.
4. gender has char(1) as a data type and not null. This attribute will be used to store the gender of the student, we only used char(1) since we only need to store 1 letter (M / F).
5. dob has date as a data type and not null. This attribute will be used to store the student’s date of birth which is why we use the date for the data type.
6. email has varchar(40) as a data type and not null. This attribute will be used to store the student’s email.
7. user\_id is foreign key with varchar(60) as a data type and not null. This attribute will be used to store the user’s id that is from the user relation.
8. major\_id is foreign key with varchar(60) as a data type and not null. This attribute will store the Major ID of the major that the student chose when they register. The major name will be stored in the Major relation.
9. batch has smallint as data type and not null. This attribute will store the student’s batch.

Lecturers have 7 attributes:

1. lecturer\_id is a primary key with varchar(60) as a data type and not null. The lecturer\_id is going to be used in the Courses relation. This attribute will be used to store the id of the lecturer.
2. first\_name has varchar(40) as a data type and not null. This attribute will be used to store the lecturer's first name.
3. last\_name has varchar(40) as a data type and not null. This attribute will be used to store the lecturer's last name.
4. gender has char(1) as a data type and not null. This attribute will be used to store the gender of the lecturer, we only used char(1) since we only need to store 1 letter (M / F).
5. dob has date as a data type and not null. This attribute will be used to store the lecturer’s date of birth which is why we use the date for the data type.
6. email has varchar(40) as a data type and not null. This attribute will be used to store the lecturer’s email.
7. user\_id is foreign key with varchar(60) as a data type and not null. This attribute will be used to store the user’s id that is from the user relation.

Course has 4 attributes:

1. course\_id is a primary key with varchar(60) as a data type and not null. This attribute is going to be in the Class relation. This attribute will store the course’s id.
2. course\_name has varchar(40) as a data type and not null. This attribute will store the name of the course.
3. lecturer\_id has varchar(60) as a data type and not null. This attribute is going to be used to store the lecturer id from the Lecturer relation.
4. scu has tinyint as a data type and not null. This attribute will be used to store the course’s scu.

Class has 4 attributes:

1. class\_id is a primary key with varchar(60) as a data type and not null. This attribute will be used to store the id of the classes.
2. Class\_name has varchar(40) as a data type and not null. This attribute will be used to store the name of the classes.
3. Course\_id is a foreign key with varchar(60) as a data type and not null. This attribute will be used to store the id of the course from the Course relation.
4. batch has smallint as a data type and not null. This attribute will be used to store the batch of the class.

Enrolled\_Courses has 4 attributes:

1. student\_id is a foreign key with varchar(60) as a data type and not null. This attribute will be used to store the student id that is enrolled in a course.
2. Class\_id is a foreign key with varchar(60) as a data type and not null. This attribute will be used to store the id of a class that is enrolled by the student.
3. mid\_score has tinyint as a data type and cannot be set to null. This attribute will store the mid exam score. The default value will be -1 until the lecturer’s update the score.
4. final\_score has tinyint as a data type and cannot be set to null. This attribute will store the final exam score. The default value will be -1 until the lecturer’s update the score and after updating, it will also calculate the gpa for the student.

Schedules has 3 attributes:

1. schedule\_id is a primary key with varchar(60) as a data type and not null. This attribute will be used to store the id of a schedule.
2. date\_time has date as data type and not null. This attribute has a date datatype so a student can check the schedule and see when the class starts.
3. class\_id is a foreign key varchar(60) as a data type and not null. This attribute will be used to tell the student what class they will be having.

Absences has 2 attributes:

1. student\_id is a foreign key with varchar(60) as a data type and not null. This attribute will be used to denote the id of the student that was absent.
2. schedule\_id is a foreign key with varchar(60) as a data type and not null. This attribute will be used to store the date and which class it was when a student is absent.

Normalisations:

User:

Fd1: user\_id -> password,role

Students:

Fd1: student\_id -> user\_id,first\_name,last\_name,gender,dob,email,batch,major\_id

Fd2: user\_id -> student\_id,first\_name,last\_name,gender,dob,email,batch,major\_id

Lecturers:

Fd1: lecturer\_id -> first\_name,last\_name,gender,dob,email,user\_id

Fd2: user\_id -> first\_name,last\_name,gender,dob,email,lecturer\_id

Major:

Fd1: major\_id -> major\_name

Courses:

Fd1: course\_id -> course\_name,lecturer\_id,scu

Class:

Fd1: class\_id -> class\_name,course\_id,batch

Enrolled\_Courses:

Fd1: student\_id, class\_id-> mid\_score,final\_score

Schedule:

Fd1: schedule\_id -> date\_time,class\_id

Absences:

No functional dependencies

Sample Queries to Generate Report:

Query to generate list of courses with the lecturer’s name and scu of a student:

**SELECT** **CONCAT**(cs.course\_name,' - ',c.class\_name) **AS** class,**CONCAT**(l.first\_name,' ',l.last\_name) **AS** teacher,cs.scu

**FROM** Enrolled\_Courses e,Class c, Courses cs, Lecturers l

**WHERE** student\_id = (**SELECT** student\_id **FROM** Students **WHERE** user\_id = ?)

**AND** e.class\_id = c.class\_id

**AND** c.course\_id = cs.course\_id

**AND** l.lecturer\_id = cs.lecturer\_id

Query to generate list of students and their scores for a class:

**SELECT DISTINCT** s.student\_id,concat(s.first\_name,' ',s.last\_name) **AS** name,e.mid\_score,e.final\_score **AS** Name

**FROM** Enrolled\_Courses e, Students s

**WHERE** e.class\_id = ?

**AND** e.student\_id=s.student\_id

Query to generate schedule of a student:

**SELECT** s.date\_time,c.class\_name,cs.course\_name

**FROM** Class c,Enrolled\_Courses e,Schedules s,Courses cs

**WHERE** e.student\_id = (**SELECT** student\_id **FROM** Students **WHERE** user\_id = ?)

**AND** e.class\_id = c.class\_id

**AND** s.class\_id = c.class\_id

**AND** cs.course\_id = c.course\_id

Query to generate student information:

**SELECT** first\_name,last\_name,gender,dob,email, (select sum(scu)

**FROM** Courses c,Enrolled\_Courses e, Class cl

**WHERE** e.student\_id = (select s.student\_id from Students s where s.user\_id = ?)

**AND** e.class\_id = cl.class\_id

**AND** cl.course\_id = c.course\_id) **AS** scu ,batch,m.major\_name

**FROM** Students s,Majors m

**WHERE** user\_id = ?

**AND** m.major\_id = s.major\_id"

Query to generate student absences:

**SELECT** c.class\_name,x.course\_name,count(x.course\_name) **AS** absent

**FROM** Absence a,Schedules s,Class c,Courses x

**WHERE** a.schedule\_id = s.schedule\_id

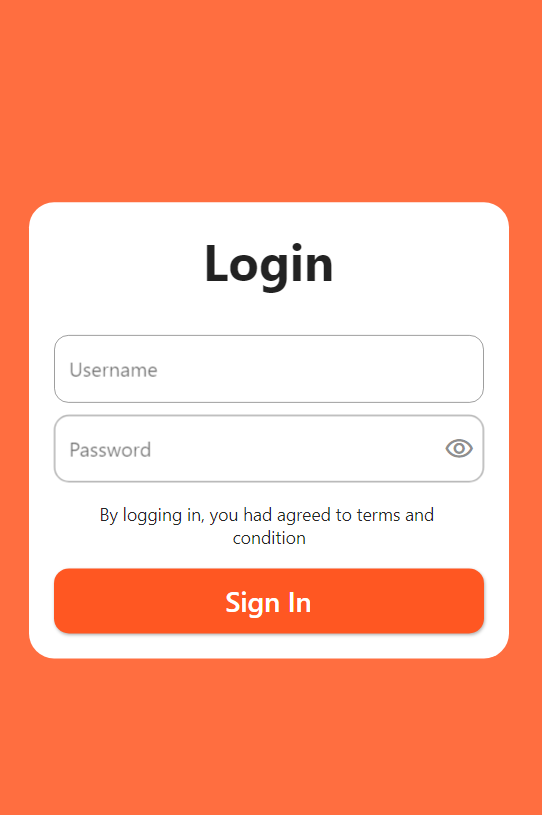
**AND** c.class\_id = s.class\_id

**AND** x.course\_id = c.course\_id

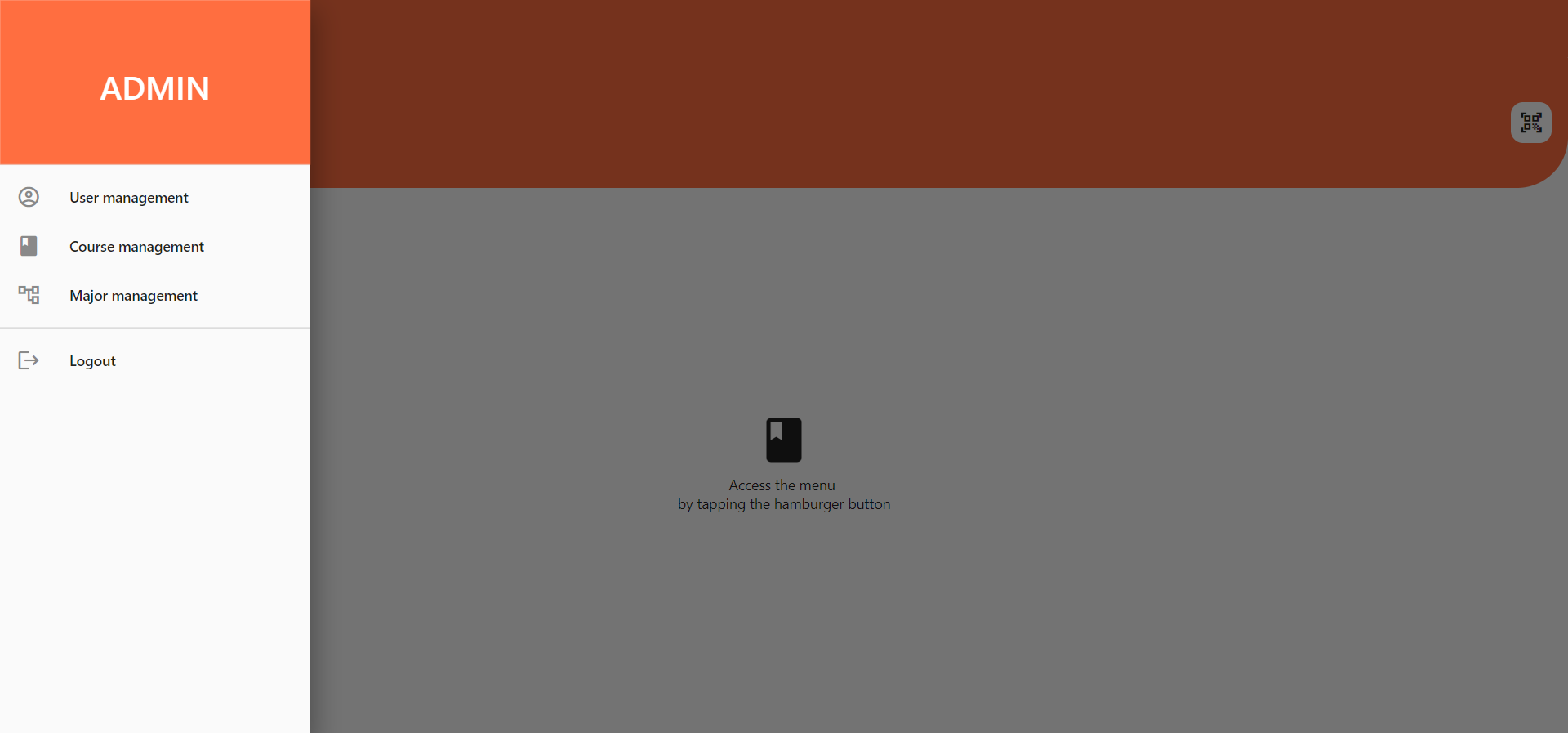
**AND** a.student\_id = (**SELECT** student\_id **FROM** Students **WHERE** user\_id = ? **LIMIT** 1)

**GROUP BY** x.course\_name, c.class\_name

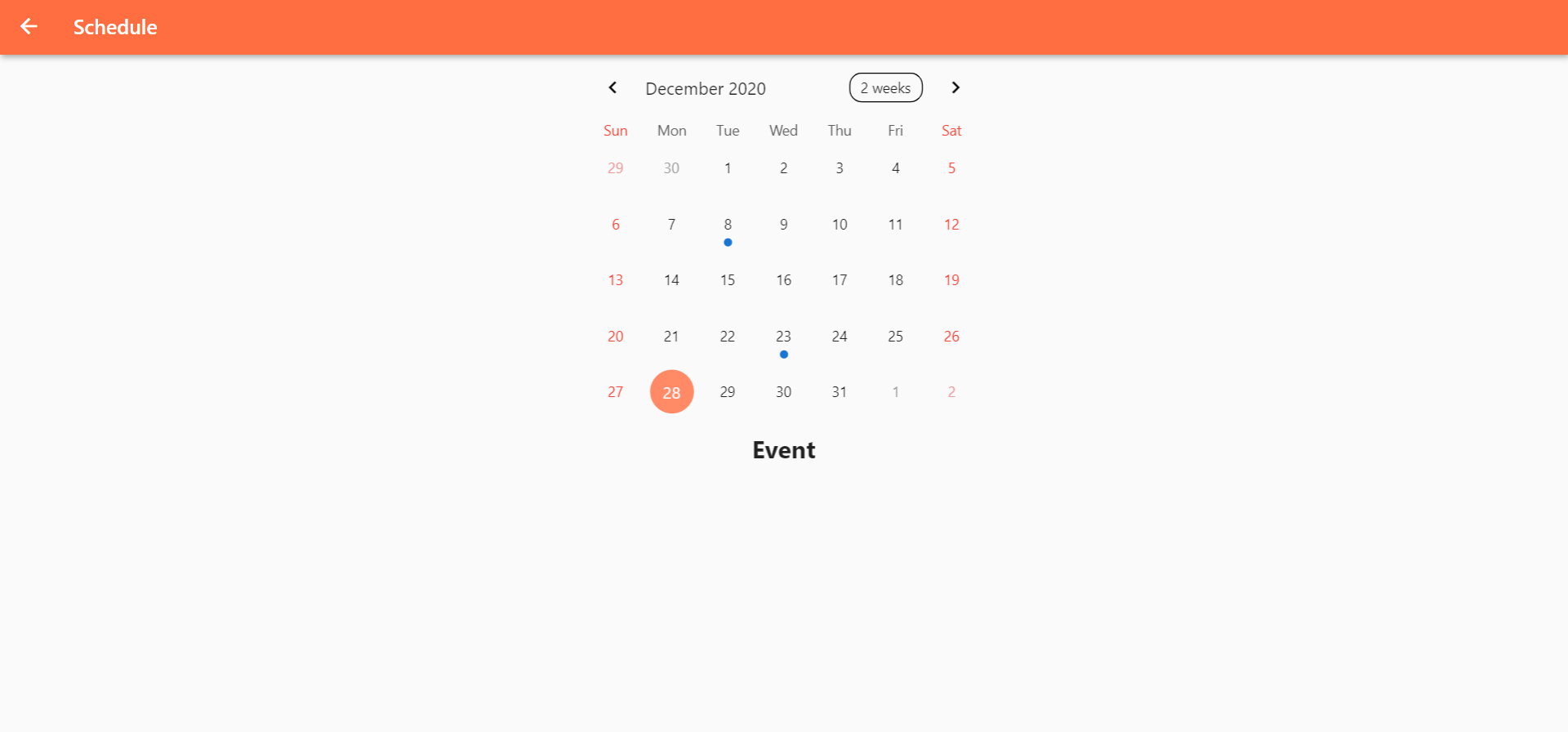
User Interfaces:



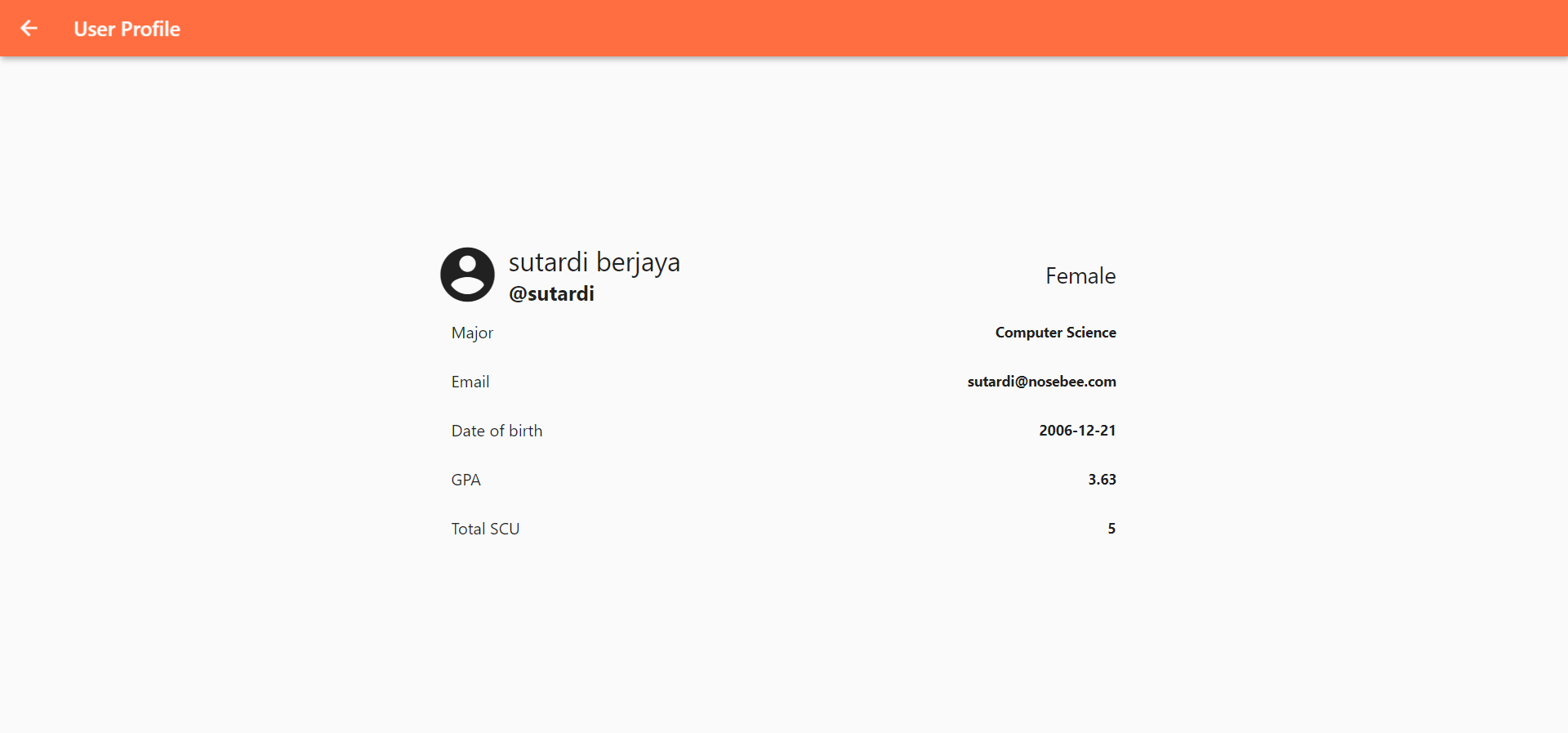
This is the login page of our app. This is where the user will insert their username and password so the app can identify who they are.



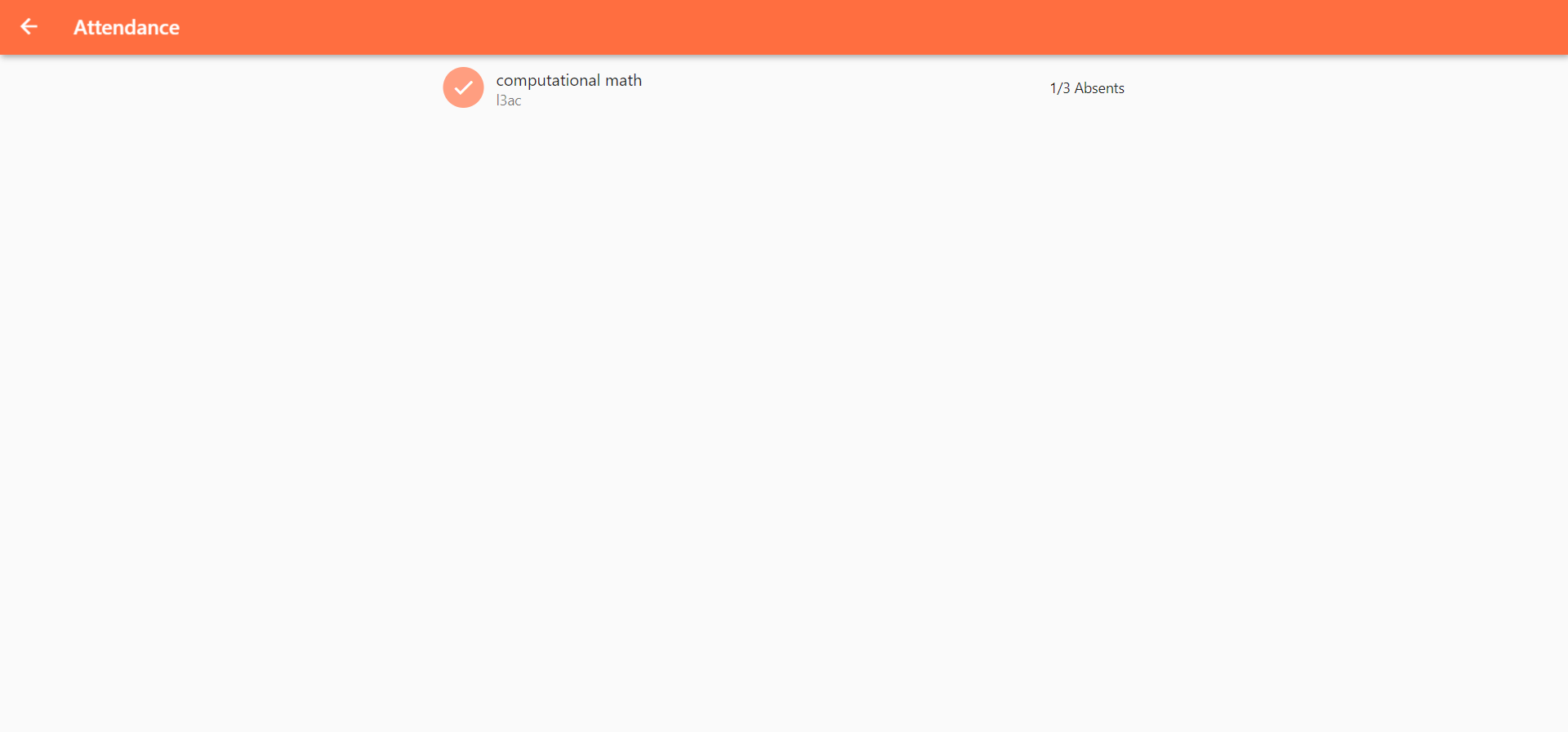
This is the main menu page of our app. Depending on the role of the user in the database, the menu shown in the tab will change accordingly.



This is the schedule section of our app. Here, students can view a calendar with highlights on dates that have a class.



This is the student info page of our app. This is where students can view all their basic information ranging from their name to their GPA and SCU.



This is the student’s absence status page of our app. Here, students can see how many absences they have for a class and the maximum absent limit on that class.

Database Security:

Our database makes use of a RESTful API. This means that the user is never directly communicating with the database thus preventing them from accessing any data they want. The client side software will instead send HTTP requests to the server such as GET, POST, PUT and DELETE. GET is equivalent to retrieving data, POST is equivalent to inserting data, PUT is equivalent to updating data and DELETE is equivalent to deleting data. This way, the user will instead interact with the server and the server will then communicate with the database.

The user access will be restricted to each role to make it more secure so the students won’t be able to access the lecturer account or an admin account and this applies for every other type of account. Different roles will have different capabilities and functionalities that are shown in the application’s interface through a panel that shows the possible actions of the user. For example, an admin is able to assign courses, majors and create users but a student or lecturer won’t be able to do the same. Even the admin does not have direct access to the contents of the database because of the RESTful API and so the limitations of a user are constrained to the user interface being provided by the application. For example, if our application does not provide a feature to delete a record in a table, an admin account could possibly have the authority to do so but would be unable to because no option to do so is available. This is used to enforce the security within the application so that only users with certain roles are able to alter the contents of the tables while also restricting full access to the database to prevent human error or tampering.

As our application will log in automatically when the user quits the app after the user has logged in for the first time, we decided to use JWT(JSON Web Tokens) to make it more secure. When a user login for the first time with their username & password, it will be verified in the API. Once it is verified, it will create a JSON Web Token and sign it using a secret key before returning it to the user’s app. The user’s app will receive and verify the token to make sure it’s authentic and then proceed to use it on every subsequent request to authenticate the user without having to send their credentials. For this project, the JSON Web Token is also encrypted to make it more secure.

Links:

<https://nosebee.lightbear.net/#/> (web-based application link)

<https://github.com/raveltan/nusbi-server> (Go Server github link)

<https://github.com/raveltan/nusbi_flutter> (Flutter Frontend github link)

<https://drive.google.com/file/d/1_KZVgsAiDUzcpr8QznqXzfLMnXwri5hz/view?usp=sharing> (video of the app demo)