CIS020-2 – Systems Development and Modern Database Practices

Assignment 2 – Student Information Kiosk (SIK) Case Study - Group Assignment

Group Number – Group 12

1814927 – Michael Songer

1802745- Zaid Rasool

Contents

[Team member roles 2](#_Toc41652447)

[Introduction 2](#_Toc41652448)

[Plan of work 2](#_Toc41652449)

[Role identification 2](#_Toc41652450)

[Users 2](#_Toc41652451)

[Client/stakeholder 3](#_Toc41652452)

[Service/audience 3](#_Toc41652453)

[Soft Systems Methodology (SSM) 3](#_Toc41652454)

[Rich Picture 3](#_Toc41652455)

[Root Definition 3](#_Toc41652456)

[CATWOE 3](#_Toc41652457)

[Conceptual Data Model 3](#_Toc41652458)

[The three E’s 3](#_Toc41652459)

[Requirements 4](#_Toc41652460)

[Use case diagram 4](#_Toc41652461)

[Activity diagram 7](#_Toc41652462)

[Class diagram 9](#_Toc41652463)

[Entity relationship diagram 10](#_Toc41652464)

[Skeleton Tables 10](#_Toc41652465)

[Entity Relationships 11](#_Toc41652466)

[In Entity relationship diagrams there are relations between student, booking, admin and event. 11](#_Toc41652467)

[Normalisation 11](#_Toc41652468)

[Data Dictionary 12](#_Toc41652469)

[Database Design 14](#_Toc41652470)

[Tables 14](#_Toc41652471)

[SQL 15](#_Toc41652472)

[User-interface design 16](#_Toc41652473)

[Security 17](#_Toc41652474)

[Implementation of kiosk 17](#_Toc41652475)

[Innovations 17](#_Toc41652476)

[Test system 17](#_Toc41652477)

[Future evolutions 17](#_Toc41652478)

[Conclusion 17](#_Toc41652479)

[References 18](#_Toc41652480)

[Appendix 18](#_Toc41652481)

# Team member roles

1814927 - Michael songer

Team manager, group report, introduction, plan of work, role identification, soft system methodology, requirements, database design, user-interface design, security, implementations of kiosk, innovations, test system, future evolutions.

1802745-Zaid Rasool

Use case diagram, activity diagram, Class diagram, Entity relationship, Normalisation, Data dictionary, Skelton tables, Entity relationship diagrams, Rich picture, Conclusion

# Introduction

The goal of the assignment is to design and make a student information kiosk that will be a useful tool for university students. It should have a variety of tools that informs students about different what’s happening both in the university and locally.

# Plan of work

The group originally started as six people but ended up going down to two, making doing the work very difficult. So, we tried to split the work evenly. Michael was set as group leader, but assigning the work was agreed together. Michael was to come up with the requirements and design the soft system methodologies. From the requirements Zaid was to do the UML diagrams. Michael then was to work on the database design and GUI design. Once all the designing was done and agreed upon it will be implemented into oracle apex.

# Role identification

## Users

Students, university staff and admin. The university staff and admin will be the ones who add things like event details and local points of interest.

## Client/stakeholder

University dean, university students, university staff, university event management team, university administration.

## Service/audience

The service is an online kiosk that is designed to provide information to students of the university of Bedfordshire. The information will be things like local places of interest, events (both on campus and near). It will also inform the students of where they can get support if they require it.

# Soft Systems Methodology (SSM)

## Rich Picture

Student information kiosk the rich picture is used to define a situation and show in different diagrams. A rich picture shares different situations. In a rich picture, they are described in two steps.

* In Student, information kiosk identifies the issue that wishes to address
* Develop an unstructured description of a situation where issues lie

A rich picture participate in the discussion and share the good situation

. (Stevens, 2016)

## Root Definition

The student information kiosk is made to easily show students anything that will be of interest to them (x), by means of a simple/easy to use online system(y), in order to aid them in getting a memorable university experience and meet other students with similar interest.

## CATWOE

C – Students

A – University staff

T – Increase popularity/knowledge of things students may be interested in.

W – Having students socialise together more will help their social skills for the future and having students at local events will increase people’s views on the university.

O – University dean and university event management team.

E – University budget, internet/mobile data connectivity, student devices, interest.

## Conceptual Data Model

## The three E’s

Efficacy -

Efficiency -

Effectiveness -

# Requirements

We were able to get an initial list of requirements from the clients brief.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Functional requirements | | | | |
| ID | Requirement | Priority | Owner | Source |
| 1.0 | Track campus events | High | Michael | Client brief |
| 1.0.1 | Bookmark campus events | Medium | Michael | Client brief |
| 1.1 | Track local events | High | Michael | Client brief |
| 1.1.1 | Bookmark local events | Medium | Michael | Client brief |
| 1.2 | Track points of interest | High | Michael | Client brief |
| 1.2.1 | Bookmark points of interest | Medium | Michael | Client brief |
| 1.3 | Track local support for students | High | Michael | Client brief |
| 1.3.1 | Bookmark local support for students | Medium | Michael | Client brief |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Non-functional requirements | | | | |
| ID | Requirement | Priority | Owner | Source |
| 2. | Accessibility options | High | Michael | Client brief |
| 3. | Efficient data usage | High | Michael | Client brief |
| 4. | Simple to use | High | Michael | Client brief |
| 5. | Minimum device storage | Medium | Michael | Client brief |

# Use case diagram

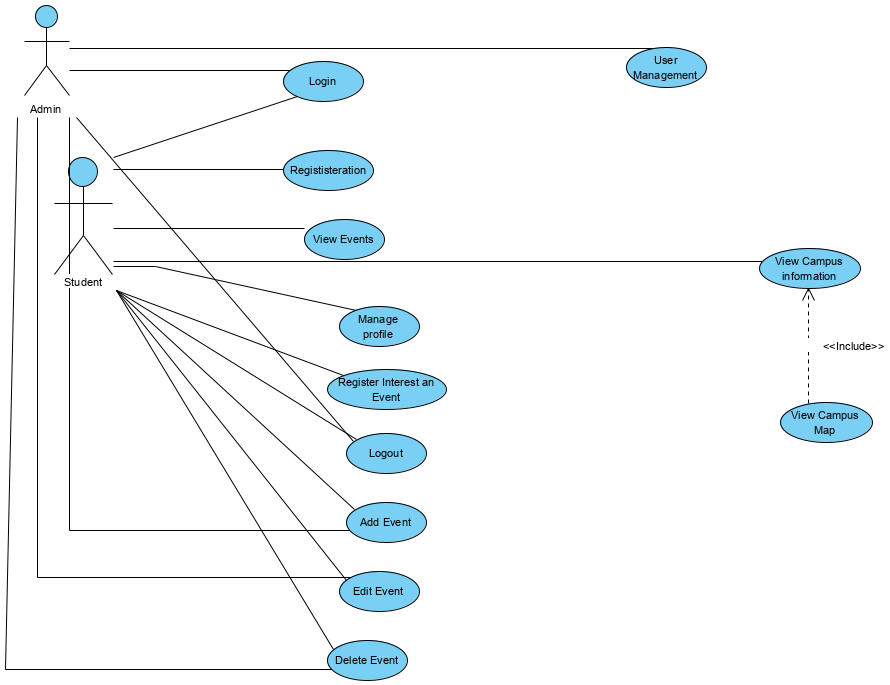
The Use case diagram between student and database are following:

A close up of a map

Description automatically generated

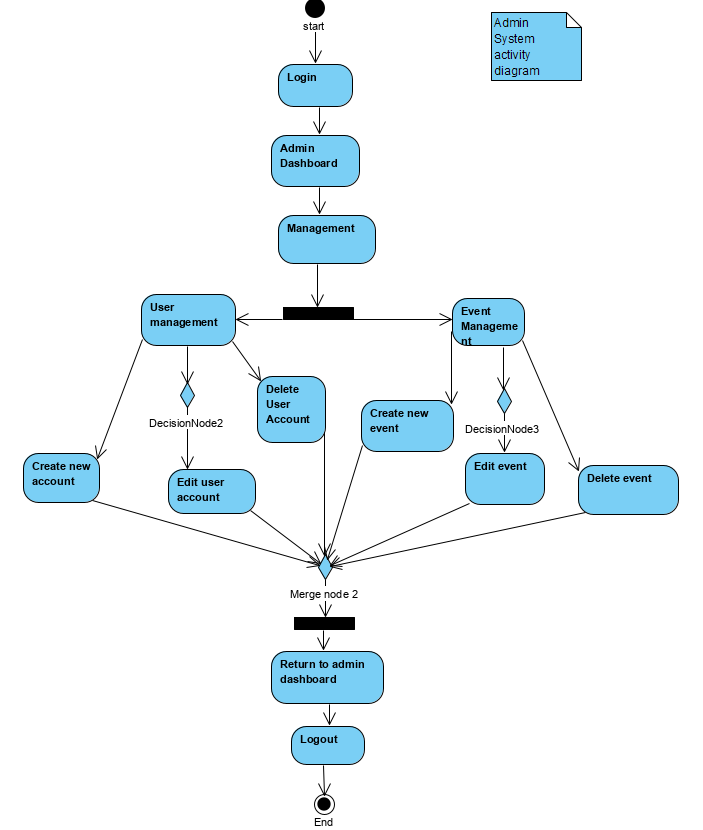
A close up of a map

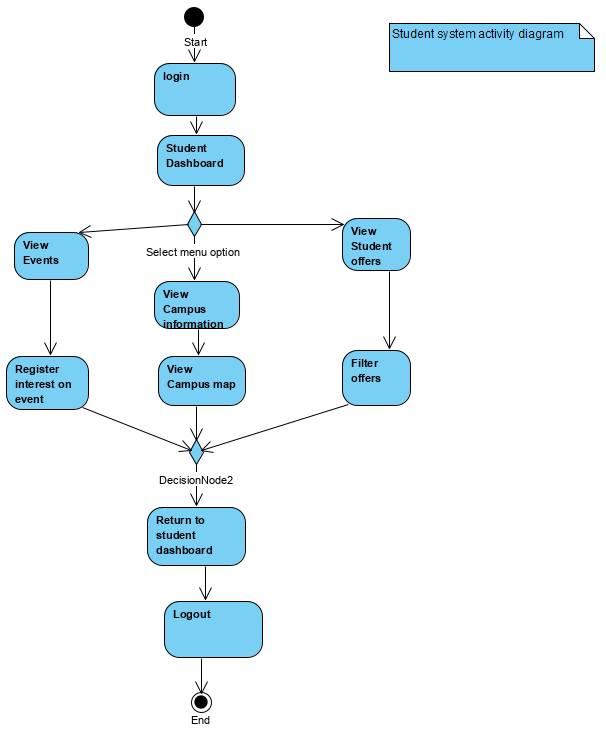
Description automatically generated

****

# Activity diagram

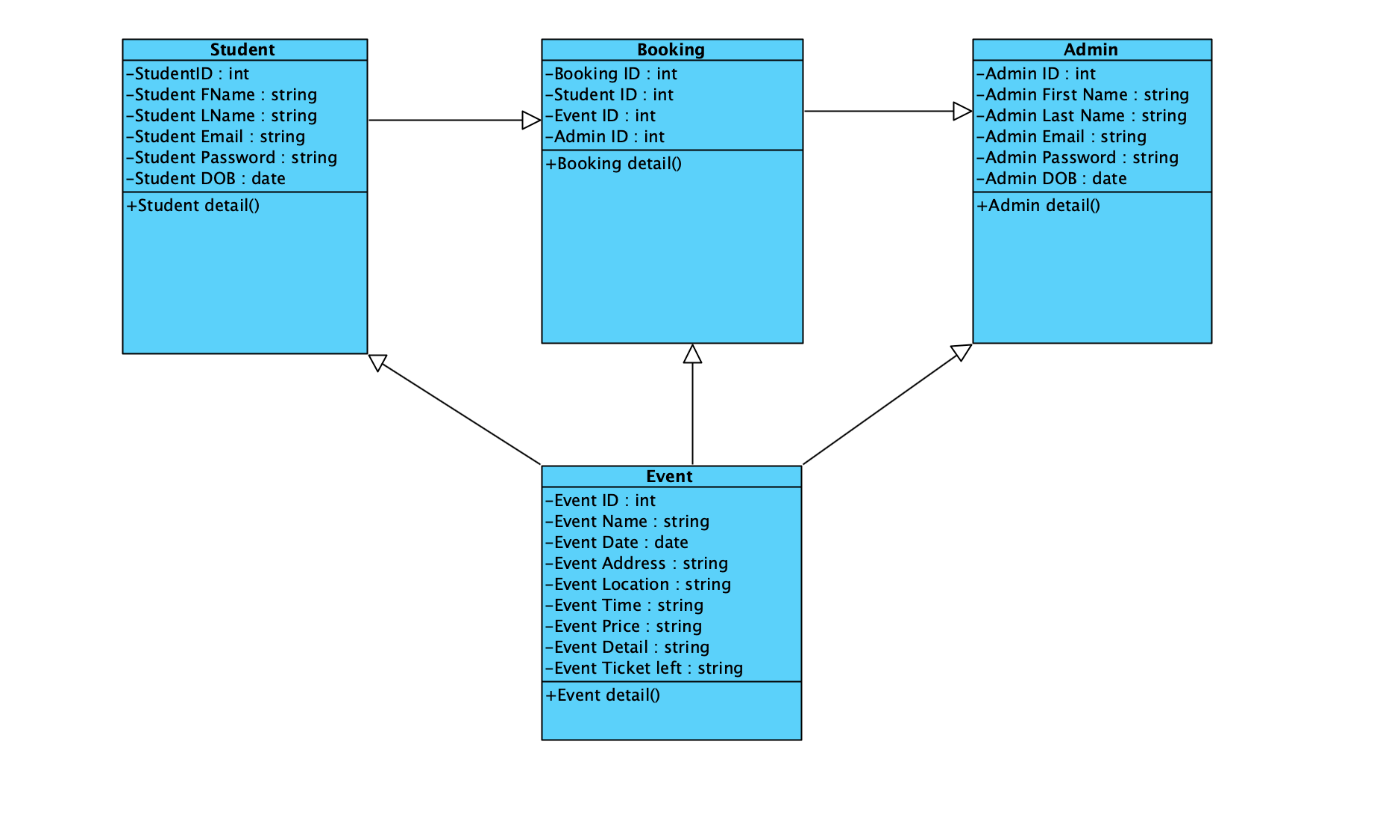
The activity diagram are following:

****



# Class diagram

The class diagram between student, booking, admin and event are following:



# Entity relationship diagram

The entity relationship between student, booking, admin and event are following:

A screenshot of a computer

Description automatically generated

# Skeleton Tables

Student(StudentID\*, Student\_FName, Student\_LName, Student\_Email, Student\_Password, Student\_DOB)

Booking(Booking\_ID\*, Student\_ID, Event-ID, Admin\_ID)

Admin(Admin\_ID\*, Admin\_FName, Admin\_LName, Admin\_Email, Admin\_Password, Admin\_DOB)

Event(Event\_ID\*, Event\_Name, Event\_Date, Event\_Address, Event\_Location, Event\_Time, Event\_Price, Event\_Detail, Event\_Ticket left)

# Entity Relationships

The entity relation between student, booking, admin and event are following:

Admin

Booking

Student

Event

# In Entity relationship diagrams there are relations between student, booking, admin and event.

1. One to one relationship
2. One to many relationship
3. Many to One relationship
4. Many to many relationship

# Normalisation

The Normalisation table are following:

|  |  |  |  |
| --- | --- | --- | --- |
| UNF | 1NF | 2NF | 3NF |
| **STD\_ID**  STD\_FName  STD\_Surname  STD\_DoB  STD\_Email  STD\_Password  STD\_Type  Event\_ID  Event\_Name  Event\_Date  Event\_Time  Event\_Location  Event\_Price  Event\_Category  Event\_Information  Tickets\_left  Booking\_ID  Booking Date  Admin\_ID  Admin\_FName  Admin\_Surname  Admin\_DoB  Admin\_Email  Admin\_Password | **STD\_ID**  STD\_FName  STD\_Surname  STD\_DoB  STD\_Email  STD\_Password  STD\_Type  **STD\_ID**  **Event\_ID**  Event\_Name  Event\_Date  Event\_Time  Event\_Location  Event\_Price  Event\_Category  Event\_Information  Tickets\_left  Booking\_ID  Admin\_ID  Admin\_FName  Admin\_Surname  Admin\_DoB  Admin\_Email  Admin\_Password | **STD\_ID**  STD\_FName  STD\_Surname  STD\_DoB  STD\_Email  STD\_Password  STD\_Type  **STD\_ID**  **Event\_ID**  Event\_Name  Event\_Date  Event\_Time  Event\_Location  Event\_Price  Event\_Category  Event\_Information  Tickets\_left    **STD\_ID**  **Booking\_ID**  **Admin\_ID**  Admin\_FName  Admin\_Surname  Admin\_DoB  Admin\_Email  Admin\_Password | **STD\_ID**  STD\_FName  STD\_Surname  STD\_DoB  STD\_Email  STD\_Password  STD\_Type  **STD\_ID**  **Event\_ID**  Event\_Name  Event\_Date  Event\_Time  Event\_Location  Event\_Price  Event\_Category  Event\_Information  Tickets\_Remaining    **STD\_ID**  **Booking\_ID\***  Event\_ID\*  **Admin\_ID**  Admin\_FName  Admin\_Surname  Admin\_DoB  Admin\_Email  Admin\_Password |

# Data Dictionary

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| S | T | U | D | E | N | T |  |
| Field Name | Datatype | Length | Index | Null | Default | Validation rule | Description |
| ID | Number(10) | 11 | PK | No |  |  | ID of student |
| FName | Varchar(10) | 11 |  | No |  |  | FName of student |
| LName | Varchar(20) | 21 |  | No |  |  | LName of student |
| Email | Varchar(30) | 31 |  | No |  | Must be sign @ | Email of student |
| Password | Varchar(40) | 41 |  | No |  |  | Password of student |
| DOB | Date(10) | 11 |  | No |  | Must be format DD/MM/YYYY | DOB of student |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| B | O | O | K | I | N | G |  |
| Field Name | Datatype | Length | Index | Null | Default | Validation rule | Description |
| BookingID | Number(10) | 9 | PK | No |  |  | ID of booking |
| StudentID | Number(20) | 19 | PK | No |  |  | ID of student |
| Event ID | Number(30) | 29 | PK | No |  |  | ID of Event |
| AdminID | Number(40) | 39 | Pk | No |  |  | ID of Admin |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A | D | M | I | N |  |  |  |
| Field Name | Datatype | Length | Index | Null | Default | Validation rule | Description |
| ID | Number(10) | 12 | Pk | No |  |  | ID of Admin |
| First Name | Varchar(10) | 22 |  | No |  |  | First Name of Admin |
| Last Name | Varchar(20) | 32 |  | No |  |  | Last Name of Admin |
| Email | Varchar(30) | 42 |  | No |  | Must be @ sign | Email of Admin |
| Password | Varchar(40) | 52 |  | No |  |  | Password of Admin |
| DOB | Date(10) | 12 |  | No |  | Must be format DD/MM/YYYY | DOB of Admin |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| E | V | E | N | T |  |  |  |
| Field Name | Datatype | Length | Index | Null | Default | Validation Rule | Description |
| ID | Number(10) | 13 | PK | No |  |  | ID of Event |
| Name | Varchar(10) | 23 |  | No |  |  | Name of Event |
| Date | Date(10) | 33 |  | No |  |  | Date of Event |
| Address | Varchar(20) | 43 |  | No |  |  | Address of Event |
| Location | Varchar(30) | 53 |  | No |  | Must be postcode | Location of Event |
| Time | Varchar(40) | 63 |  | No |  |  | Time of Event |
| Price | Varchar(50) | 73 |  | No |  |  | Price of Event |
| Detail | Varchar(60) | 83 |  | No |  |  | Detail of Event |
| Ticket left | Varchar(70) | 93 |  | No |  |  | Ticket left of Event |

# Database Design

## Tables

Below are the tables that will be in the database. It has been normalised up to 3NF and only stores information that is necessary to save storage.

Campus\_events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | Field size | Data Format | Key |
| Id | NUMBER | Dynamic |  | PRIMARY KEY |
| Name | VARCHAR2 | 200 |  |  |
| Start\_timestamp | TIMESTAMP |  | DD:MM:YYYY:HH24:MI:SS |  |
| End\_timestamp | TIMESTAMP |  | DD:MM:YYYY:HH24:MI:SS |  |
| Location | VARCHAR2 | 1000 |  |  |
| Details | VARCHAR2 | 4000 |  |  |

Local\_events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | Field size | Data Format | Key |
| Id | NUMBER | Dynamic |  | PRIMARY KEY |
| Name | VARCHAR2 | 200 |  |  |
| Start\_timestamp | TIMESTAMP |  | DD:MM:YYYY:HH24:MI:SS |  |
| End\_timestamp | TIMESTAMP |  | DD:MM:YYYY:HH24:MI:SS |  |
| Location | VARCHAR2 | 1000 |  |  |
| Details | VARCHAR2 | 4000 |  |  |

Local\_support

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | Field size | Data Format | Key |
| Name | VARCHAR2 | 200 |  | PRIMARY KEY |
| Location | VARCHAR2 | 1000 |  |  |
| Details | VARCHAR2 | 4000 |  |  |

Local\_poi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | Field size | Data Format | Key |
| Name | VARCHAR2 | 200 |  | PRIMARY KEY |
| Location | VARCHAR2 | 1000 |  |  |
| Details | VARCHAR2 | 4000 |  |  |

All the tables above have a corresponding bookmark table that is used to link users with the data that they have bookmarked.

Bookmarked\_campus\_events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | Field size | Data Format | Key |
| Username | NUMBER | Dynamic |  | PRIMARY KEY |
| Campus\_event\_id | NUMBER | Dynamic |  | FOREIGN KEY |

Bookmarked\_local\_events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | Field size | Data Format | Key |
| Username | NUMBER | Dynamic |  | PRIMARY KEY |
| Local\_event\_id | NUMBER | Dynamic |  | FOREIGN KEY |

Bookmarked\_local\_support

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | Field size | Data Format | Key |
| Username | NUMBER | Dynamic |  | PRIMARY KEY |
| Local\_support\_name | VARCHAR2 | 200 |  | FOREIGN KEY |

Bookmarked\_local\_poi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field name | Data type | Field size | Data Format | Key |
| Username | NUMBER | Dynamic |  | PRIMARY KEY |
| Local\_poi\_name | VARCHAR2 | 200 |  | FOREIGN KEY |

## SQL

Below is the SQL code used in order to make the tables shown above.

|  |  |
| --- | --- |
| -- Make campus events table  CREATE TABLE campus\_events(  id NUMBER NOT NULL CONSTRAINT campus\_events\_id\_pk PRIMARY KEY,  name VARCHAR2(200) NOT NULL,  start\_timestamp TIMESTAMP NOT NULL,  end\_timestamp TIMESTAMP NOT NULL,  location VARCHAR2(1000) NOT NULL,  details VARCHAR2(4000) NOT NULL  );  -- Make sequence for trigger later  CREATE SEQUENCE campus\_events\_id\_sequence;  -- Trigger to do incerement  CREATE OR REPLACE TRIGGER campus\_events\_on\_insert  BEFORE INSERT ON campus\_events  FOR EACH ROW  BEGIN  SELECT campus\_events\_id\_sequence.nextval  INTO :new.id  FROM dual;  END;  -- Make bookmarked campus event table  CREATE TABLE bookmarked\_campus\_events(  username NUMBER NOT NULL CONSTRAINT student\_id\_bookmark\_campus\_events\_pk PRIMARY KEY,  campus\_event\_id NUMBER,  CONSTRAINT campus\_event\_id\_fk FOREIGN KEY (campus\_event\_id) REFERENCES campus\_events(id)  ); | -- Make local events table  CREATE TABLE local\_events(  id NUMBER NOT NULL CONSTRAINT local\_events\_id\_pk PRIMARY KEY,  name VARCHAR2(200) NOT NULL,  start\_timestamp TIMESTAMP NOT NULL,  end\_timestamp TIMESTAMP NOT NULL,  location VARCHAR2(1000) NOT NULL,  details VARCHAR2(4000) NOT NULL  );  -- Make sequence for trigger later  CREATE SEQUENCE local\_events\_id\_sequence;  -- Trigger to do incerement  CREATE OR REPLACE TRIGGER local\_events\_on\_insert  BEFORE INSERT ON local\_events  FOR EACH ROW  BEGIN  SELECT local\_events\_id\_sequence.nextval  INTO :new.id  FROM dual;  END;  -- Make bookmarked local event table  CREATE TABLE bookmarked\_local\_events(  username NUMBER NOT NULL CONSTRAINT student\_id\_bookmark\_local\_events\_pk PRIMARY KEY,  local\_event\_id NUMBER,  CONSTRAINT local\_event\_id\_fk FOREIGN KEY (local\_event\_id) REFERENCES local\_events(id)  ); |
| -- Make local support table  CREATE TABLE local\_support(  name VARCHAR2(200) NOT NULL CONSTRAINT support\_pk PRIMARY KEY,  location VARCHAR2(1000) NOT NULL,  details VARCHAR2(4000) NOT NULL  );  -- Make bookmarked support table  CREATE TABLE bookmarked\_support(  username NUMBER NOT NULL CONSTRAINT student\_id\_bookmark\_support\_pk PRIMARY KEY,  local\_support\_name VARCHAR2(200),  CONSTRAINT support\_name\_fk FOREIGN KEY (local\_support\_name) REFERENCES local\_support(name)  ); | -- Make points of interest table  CREATE TABLE local\_poi(  name VARCHAR2(200) NOT NULL CONSTRAINT local\_poi\_id\_pk PRIMARY KEY,  location VARCHAR2(1000) NOT NULL,  details VARCHAR2(4000) NOT NULL  );  -- Make bookmarked point of interest table  CREATE TABLE bookmarked\_poi(  username NUMBER NOT NULL CONSTRAINT student\_id\_bookmark\_poi\_pk PRIMARY KEY,  local\_poi\_name VARCHAR2(200),  CONSTRAINT poi\_name\_fk FOREIGN KEY (local\_poi\_name) REFERENCES local\_poi(name)  ); |

# User-interface design

The client has said that the system should be easy to use. In order to do this the design should be a simple and as straight forward as possible. The client should also work on multiple platforms and devices, so the front-end design should be designed around multiple devices instead of just one.

Home Page

The home page is designed to be as simple as possible. In the top left is a navigation menu button that will open the navigation menu on the left with buttons to all the different pages. The buttons to the pages are also on the main page, three wide and how ever many down, depending on the need. Having the buttons like this mean that no matter the size of the page they will be easy to see and use.

All other pages will link back to the homepage and the navigation bar will be accessible everywhere, so people should not get lost within the app.

The other pages will mainly be a list of buttons that when clicked will display more information and have a button to bookmark for later.

# Security

The software has multiple types of users, all with different levels of security. Doing this means that only certain people will be able to access certain things, making the app secure. These user levels have been described earlier, so I will not go over them again, but the main thing is these levels can be added to and taken away depending on the progression of the software.

# Implementation of kiosk

Due to the time constraints and the issues within the group we have been unable to make a working example of the kiosk. If given more time and maybe more people, the work we have done could be made into a working software.

# Test system

We are not able to test the system because it does not exist but if we were to test it we would use a variety of tests, including test cases and having other people test the software and fill out a questionnaire of their experience.

The test cases would be used in order to make sure that there are no major issues with the software, it will also test for common minor issues that may happen.

Once the test cases are done the test users will include university students, university lecturers and the university event management team.

# Future evolutions

After the tests were completed, we would then look into the future of the software. The software would evolve depending on what the stakeholders desire. This would be determined by having the users fill out questionnaires/give their opinions on how the software is already running and what they think can be added to improve it.

# Conclusion

In conclusion, the student information kiosk project was done with different ideas. Our group started with six people but ended with two (Zaid and Michael) because of a variety of things including: people not responding to messages, peoples schedule and issues caused by the coronavirus. These issues caused a lot of time to be wasted, making the project very difficult to complete to the desired standard. We did what we could given the constrains and are happy with the work we have done but wished we had more time/a better group so we could get more done.

# Appendix