Eastern Mediterranean University

Department of Computer Engineering

Software Engineering Program

Famagusta, North Cyprus

Intermediate Report

Software Design Specification Document (SDS)

CMSE 322

EMU Student Kit Android Application

Students:

1. Talal Hatem Mahdy -147139
2. Abdoulgwad Hussien Elsheredi -147597
3. Adham Moshasha – 148387
4. Mohamed M. M. Balto - 147697

Instructor: Asst. Prof. Dr. Duygu Çelik Ertuğrul
Lab Coordinator: Begüm Koru

Revision History

Date	Version	Description	Author
April 25, 2017	1.0	Software Design Specification Document	Talal, Abdoulgwad, Adham, Mohamed
May 2, 2017	1.1	Some fixes in the SRS and SDS reports	Talal, Abdoulgwad, Adham, Mohamed

April 2017

Table of Contents:

1. Introduction
    1.1. Purpose
    1.2. Scope
    1.3. Abbreviations
2. References
3. Decomposition Description
    3.1. Module decomposition
    3.2. Concurrent Process Decomposition
    3.3. Data Decomposition
4. Dependency Description
    4.1. Intermodule dependencies
    4.2. Data dependencies
5. Interface Description
    5.1. Login
    5.2. Main Interface
    5.3. Time Tables
    5.4. Bus Schedule
    5.5. News
    5.6. Notes
    5.7. Settings
    5.8. Calendar
    5.9. Map
    5.10. Info
    5.11. GPA Calculator
6. Detailed Design
    6.1. Module Detailed Design
    6.2. Data Detailed Design
    6.3. UML Interaction Diagram

1. Introduction:

1.1. Purpose:

Up to this point, a proposal and a plan to develop an application called EMU Student Kit has been made. The analysis and formation of a Software Requirements Specification (SRS) Document has also been made. This document is known as the Software Design Specification (SDS) Document[1] and it is an important document to describe all the essential components of the system in detail, satisfying the requirements given by the customer. This document will be used in the implementation stage by the developers.

1.2. Scope:

The EMU Student Kit is a complex multi-functional Android Mobile Application designed for EMU with a variety of different functions and features that are beneficiary for students. This application can also be used by any other student and not only EMU students. This project was started due to shortage of productivity and organizational applications that are specifically designed for students. Since there are over 20,000 students in EMU, this project aims to improve the performance and time management of students, give them useful information about the University, assist them during lectures, and provide many other beneficiary features for students.

1.3. Abbreviations:

SDS: Software Design Specification Document
SRS: Software Requirements Specification Document
DB: Database
UML: Unified Modeling Language
TUBITAK: The Scientific and Technological Research Council of Turkey
BPMN: Business Process Modeling Notation
GUI: Graphical User Interface
API: Application Program Interface
UI: User Interface
EMU: Eastern Mediterranean University
IEEE: Institute of Electrical and Electronics Engineers
ERD: Entity Relationship Diagram
BPMN: Business Process Model Notation

References

[1]. *IEEE Recommended Practice for Software Design Descriptions*. (1998). *Ieeexplore.ieee.org*. Retrieved 1 May 2017, from http://ieeexplore.ieee.org/document/741934

[2]. *Software Design Tools for Agile Teams*. *Visual-paradigm.com*. Retrieved 1 May 2017, from https://www.visual-paradigm.com/

[3]. *Flowchart Maker & Diagramming Software, Microsoft Visio*. *Products.office.com*. Retrieved 1 May 2017, from https://products.office.com/en-us/visio

[4]. *Gliffy | Online Diagram and Flowchart Software*. *Gliffy.com*. Retrieved 1 May 2017, from https://www.gliffy.com/

[5]. *MockFlow - Online Wireframe and UX Tools*. *Mockflow.com*. Retrieved 1 May 2017, from https://mockflow.com/

[6]. *GenMyModel*. *GenMyModel*. Retrieved 1 May 2017, from https://www.genmymodel.com/

[7]. *Timetable Management System (Activity Diagram (UML)) | Creately*. (2014). *Creately.com*. Retrieved 1 May 2017, from https://creately.com/diagram/example/hrsmhvuz1/

[8]. *UML Notetaking App (Activity Diagram (UML)) | Creately*. (2016). *Creately.com*. Retrieved 1 May 2017, from https://creately.com/diagram/example/iln4zy543/UML+Notetaking+App

[9]. Zeeshan. (2012). *Time Table Management System*. *Slideshare.net*. Retrieved 1 May 2017, from https://www.slideshare.net/imdzeeshan/time-table-mgt-system

3. Decomposition and Description:

Each entity of the system is determined and briefly described in this Section. For each entity, a detailed description will be provided in Section 6. An overall design diagram will be shown in Figure 1. This diagram was created using the Visual Paradigm Tool[2].

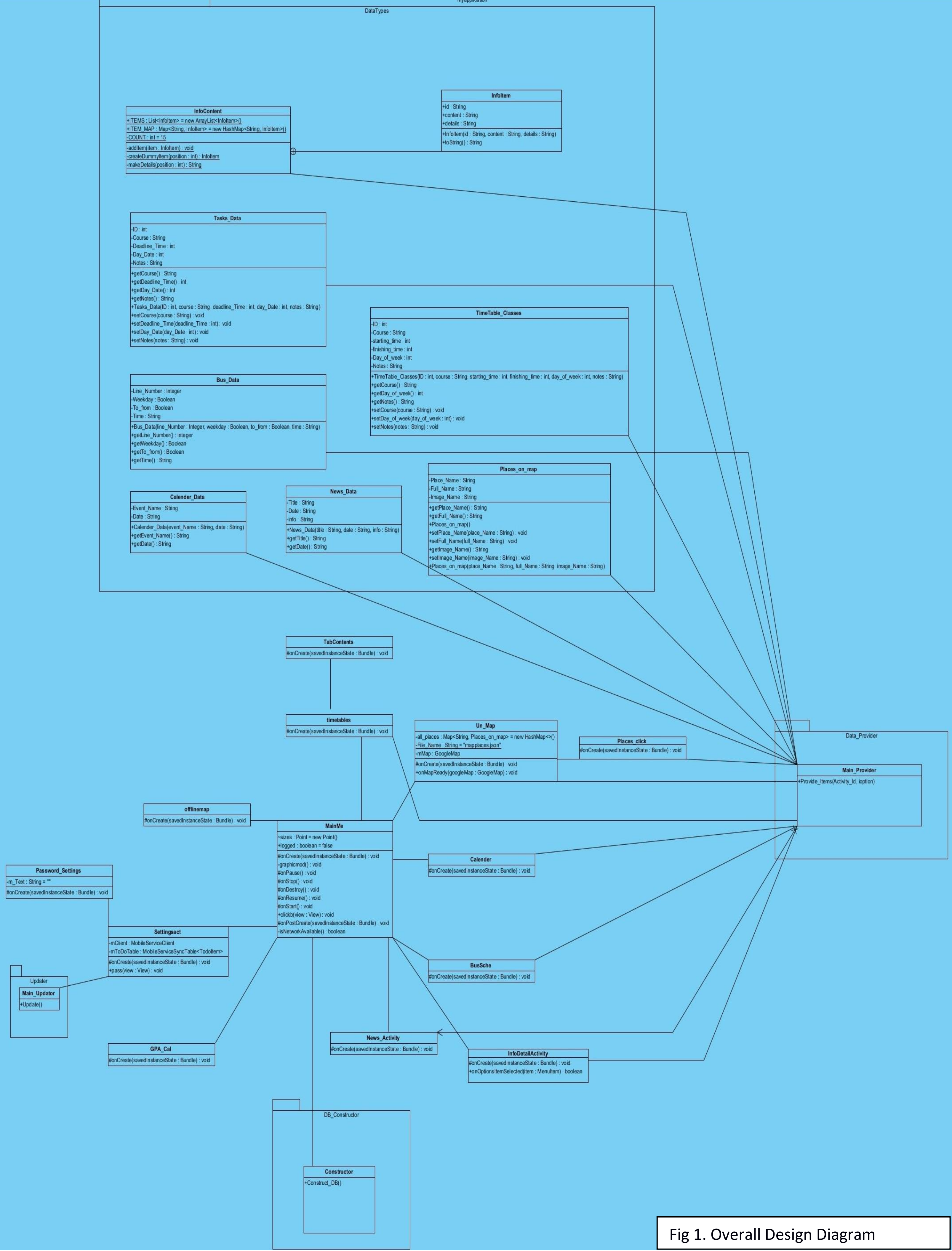

myapplication
DataTypes
InfoItem
+id : String
+content : String
+details : String
+InfoItem(id : String, content : String, details : String)
+toString() : String
InfoContent
+ITEMS : List<InfoItem> = new ArrayList<InfoItem>()
+ITEM_MAP : Map<String, InfoItem> = new HashMap<String, InfoItem>()
-COUNT : int = 15
-addItem(item : InfoItem) : void
-createDummyItem(position : int) : InfoItem
-makeDetails(position : int) : String
Tasks_Data
-ID : int
-Course : String
-Deadline_Time : int
-Day_Date : int
-Notes : String
+getCourse() : String
+getDeadline_Time() : int
+getDay_Date() : int
+getNotes() : String
+Tasks_Data(ID : int, course : String, deadline_Time : int, day_Date : int, notes : String)
+setCourse(course : String) : void
+setDeadline_Time(deadline_Time : int) : void
+setDay_Date(day_Date : int) : void
+setNotes(notes : String) : void
TimeTable_Classes
-ID : int
-Course : String
-starting_time : int
-finishing_time : int
-Day_of_week : int
-Notes : String
+TimeTable_Classes(ID : int, course : String, starting_time : int, finishing_time : int, day_of_week : int, notes : String)
+getCourse() : String
+getDay_of_week() : int
+getNotes() : String
+setCourse(course : String) : void
+setDay_of_week(day_of_week : int) : void
+setNotes(notes : String) : void
Bus_Data
-Line_Number : Integer
-Weekday : Boolean
-To_from : Boolean
-Time : String
+Bus_Data(line_Number : Integer, weekday : Boolean, to_from : Boolean, time : String)
+getLine_Number() : Integer
+getWeekday() : Boolean
+getTo_from() : Boolean
+getTime() : String
Places_on_map
-Place_Name : String
-Full_Name : String
-Image_Name : String
+getPlace_Name() : String
+getFull_Name() : String
+Places_on_map()
+setPlace_Name(place_Name : String) : void
+setFull_Name(full_Name : String) : void
+getImage_Name() : String
+setImage_Name(image_Name : String) : void
+Places_on_map(place_Name : String, full_Name : String, image_Name : String)
Calender_Data
-Event_Name : String
-Date : String
+Calender_Data(event_Name : String, date : String)
+getEvent_Name() : String
+getDate() : String
News_Data
-Title : String
-Date : String
-info : String
+News_Data(title : String, date : String, info : String)
+getTitle() : String
+getDate() : String
TabContents
#onCreate(savedInstanceState : Bundle) : void
timetables
#onCreate(savedInstanceState : Bundle) : void
Un_Map
-all_places : Map<String, Places_on_map> = new HashMap<>()
-File_Name : String = "mapplaces.json"
-mMap : GoogleMap
#onCreate(savedInstanceState : Bundle) : void
+onMapReady(googleMap : GoogleMap) : void
Places_click
#onCreate(savedInstanceState : Bundle) : void
Data_Provider
Main_Provider
+Provide_Items(Activity_Id, ioption)
offlinemap
#onCreate(savedInstanceState : Bundle) : void
MainMe
~sizes : Point = new Point()
+logged : boolean = false
#onCreate(savedInstanceState : Bundle) : void
-graphicmod() : void
#onPause() : void
#onStop() : void
#onDestroy() : void
#onResume() : void
#onStart() : void
+clickb(view : View) : void
#onPostCreate(savedInstanceState : Bundle) : void
-isNetworkAvailable() : boolean
Calender
#onCreate(savedInstanceState : Bundle) : void
Password_Settings
-m_Text : String = ""
#onCreate(savedInstanceState : Bundle) : void
Settingsact
-mClient : MobileServiceClient
-mToDoTable : MobileServiceSyncTable<TodoItem>
#onCreate(savedInstanceState : Bundle) : void
+pass(view : View) : void
Updater
Main_Updator
+Update()
BusSche
#onCreate(savedInstanceState : Bundle) : void
GPA_Cal
#onCreate(savedInstanceState : Bundle) : void
News_Activity
#onCreate(savedInstanceState : Bundle) : void
InfoDetailActivity
#onCreate(savedInstanceState : Bundle) : void
+onOptionsItemSelected(item : MenuItem) : boolean
DB_Constructor
Constructor
+Construct_DB()

Fig 1. Overall Design Diagram

3.1. Module Decomposition:

There are 4 main components in the EMU Student Kit. They are shown in Figure 2. Furthermore, we will divide our entities into three main groups which are module entities, concurrent process entities and data entities.

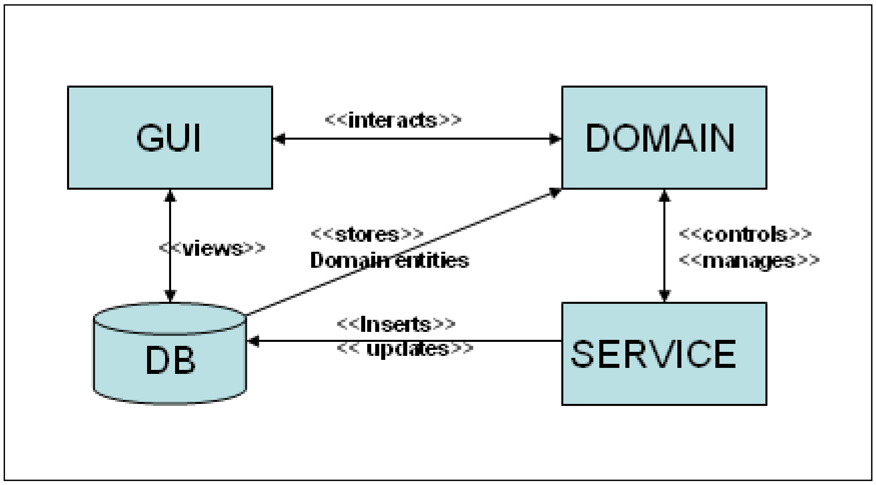


Fig 2. Components of EMU Student Kit

GUI: The Graphical User Interface is an essential and important component where the users interact with the system. We can also interact with the Domain and Service components using the Graphical User Interface component. Detailed User Interface design will be shown in Section 5.

Service: This component is an important component where its entities can manage and control other domain entities. Its entities are mainly responsible for controlling the business logic of the system. The Main_Provider and Main_Updater classes are included in this component.

Domain: The domain entities include all the other fundamental non-service entities that represent the domain objects. Also, each domain component is connected to the GUI and the local user database where its views are represented in the GUI and stored in the local user database. News_Activity class, InfoDetailActivity class, timetables class, etc are some of the domain entities.

Database: The external database component of the EMU Student Kit consists of two main databases, a local user database which stores all the data required by the system and the user, and a local resource database (online) where the administrator keeps the data about the modules updated.

The module entities of the system are written below. Some of the detailed descriptions will be provided in Section 6.

Module Entities:

3.1.1. MainMe: The entity that manages the main menu and Android button clicks activities.
3.1.2. timetables: The entity that manages the Time Tables and Tasks module.
3.1.3. Calendar: The entity that manages the university calendar module.
3.1.4. BusSche: The entity that manages the bus schedules module.
3.1.5. News_Activity: The entity that manages the university news module.
3.1.6. GPA_cal: The entity responsible for calculating the GPA/CGPA.
3.1.7. Settingsact: The entity responsible for managing many of the system's settings.
3.1.8. Constructor: The entity responsible for creating a database when it is not available.
3.1.9. Un_Map: The entity responsible for managing the interactive map.
3.1.10. Password_Settings: The entity responsible for managing the PIN code.

3.2. Concurrent Process Decomposition:

3.2.1. Main Provider: retrieves the data from the local user database.
3.2.2. Updater: Updates the local Database using the online database.

3.3. Data Decomposition:

The data entities are mentioned here. Also, a context diagram, a level 0 and a level 1 Data Flow Diagrams (for Time Tables) are shown in Figures 3 to 5. These Figures were created using Microsoft Visio Tool[3].

3.3.1. Places_on_map: entity holding all the details of the places on the interactive map.
3.3.2. News_Data: entity containing all the university news details.
3.3.3. Calendar_Data: entity containing all the university calendar details.
3.3.4. Bus_Data: entity containing all the bus related data.
3.3.5. TimeTable_Classes: entity containing the time table related data.
3.3.6. Tasks_Data: entity that contains the tasks related data.
3.3.7. InfoContent: entity that creates some sample university related information.
3.3.8. Infoitems: entity that contains the university related information.

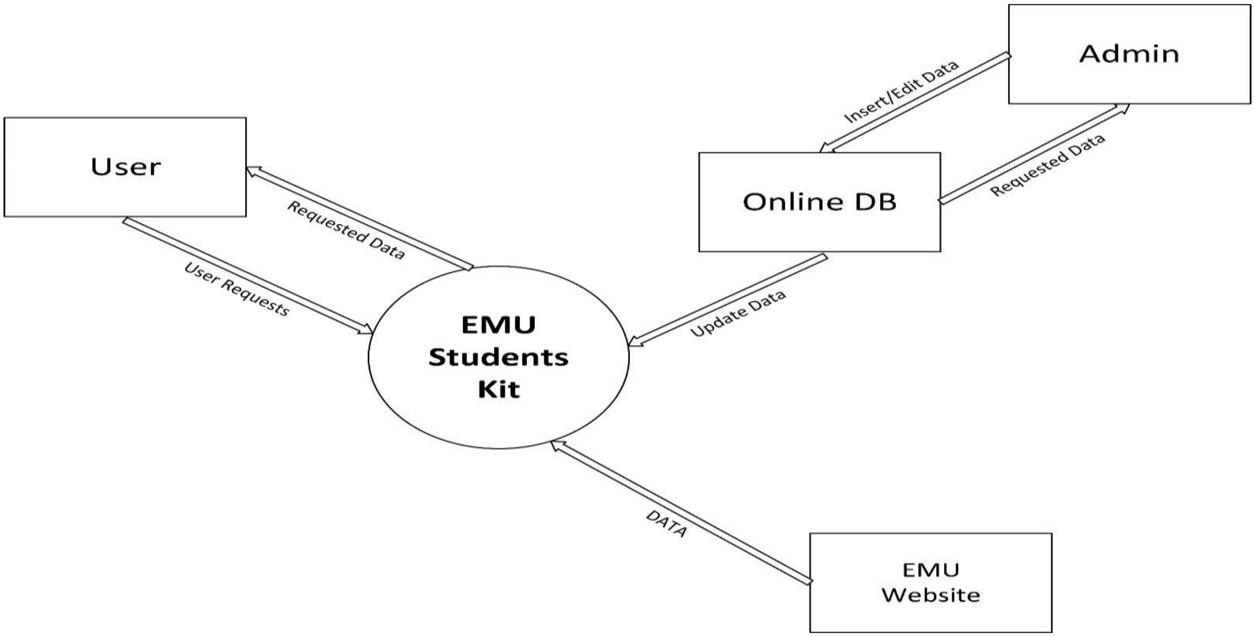


Fig 3. Context Diagram

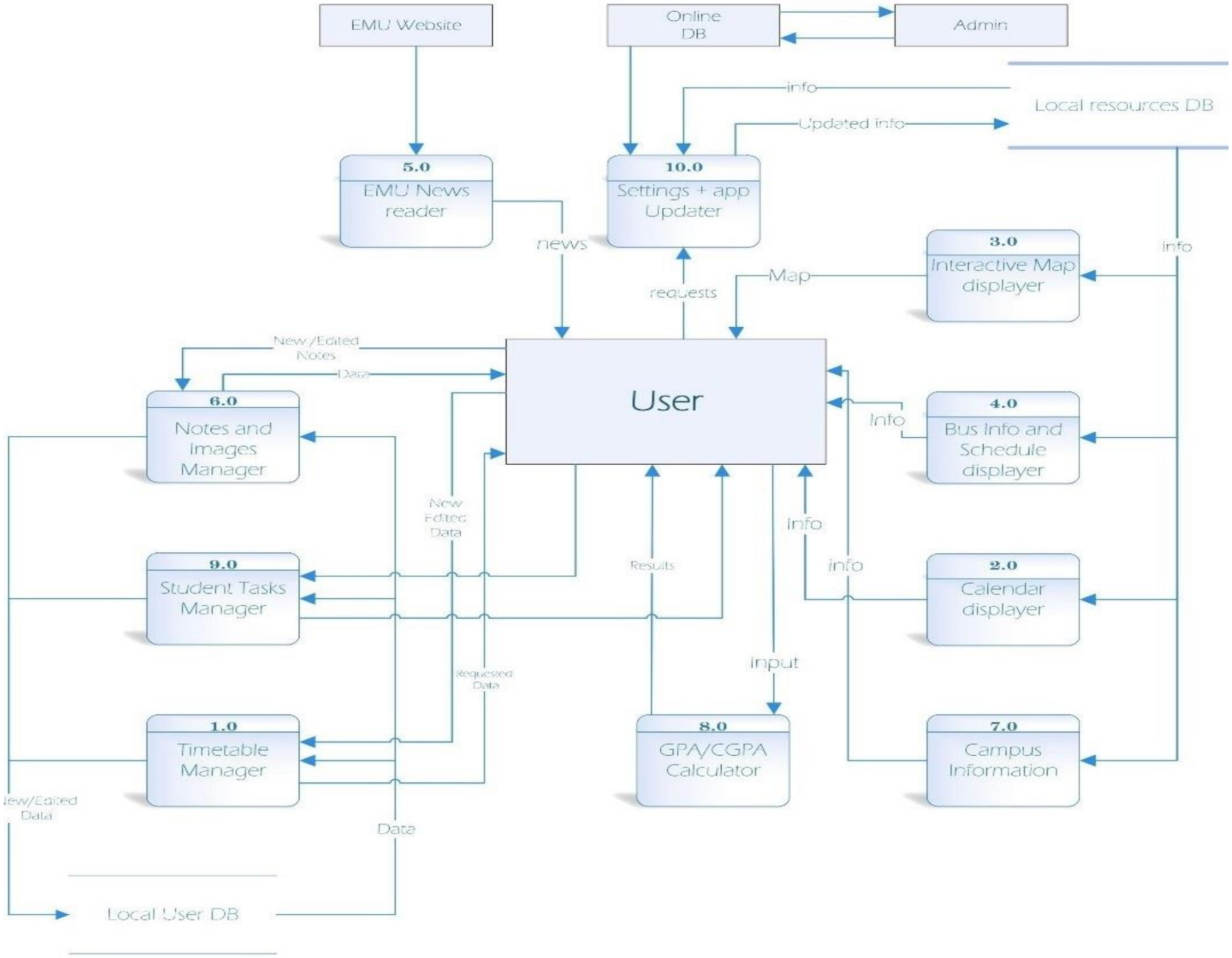


Fig 4. Level 0 Data Flow Diagram

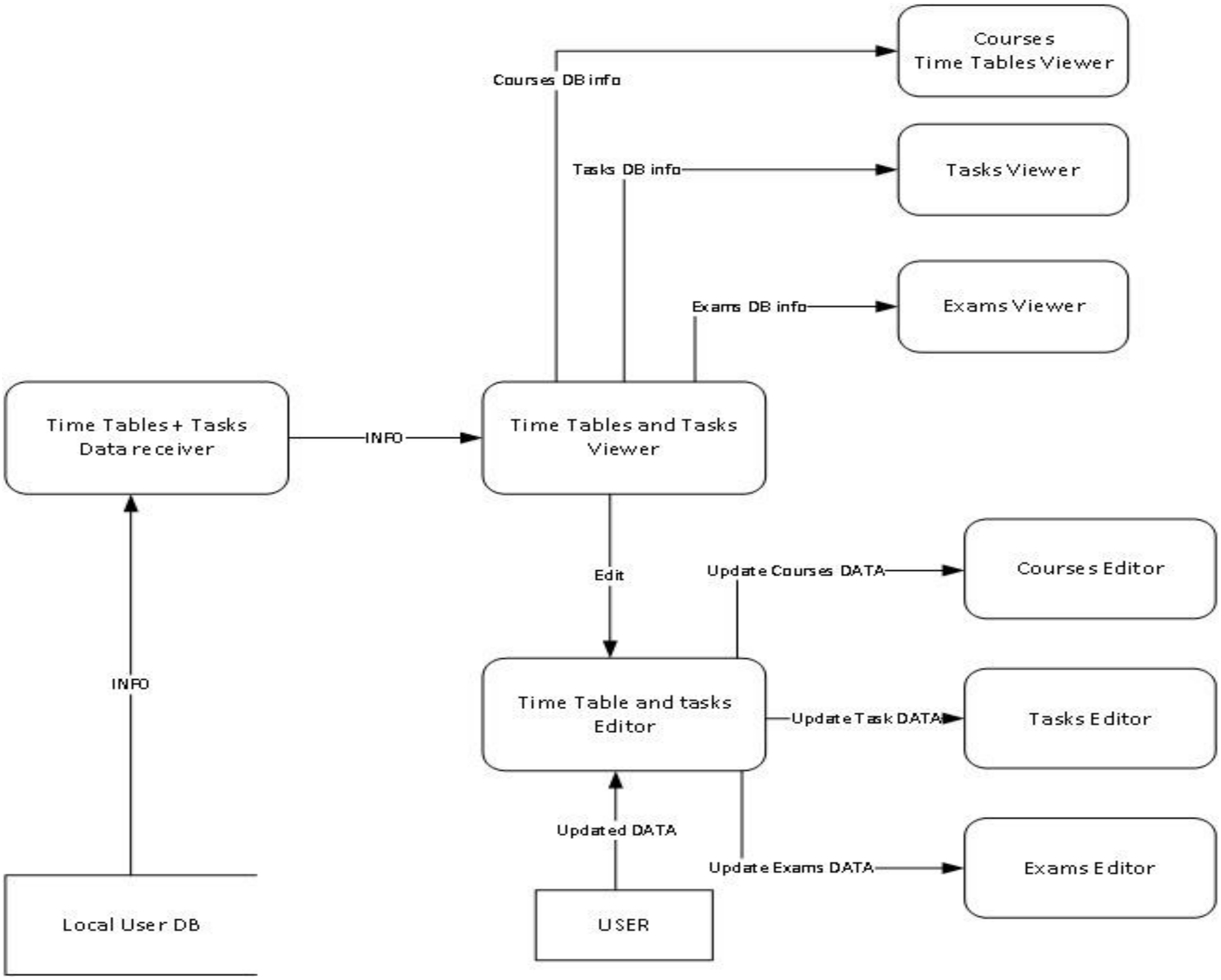


Fig 5. Level 1 Data Flow Diagram (Time Tables)

4. Dependency Description:

4.1. Intermodule Dependencies:

As indicated in Figure 2, users depend on the GUI to access all the domain modules. The service module acts as a medium between the database and the domain module where it edits/updates the database data and manages some of the domain modules. Also, the domain modules get their data from the database.

4.2. Data Dependencies:

This will be represented as an Entity Relationship Diagram in Figure 6. This diagram was created using the Gliffy tool[4].

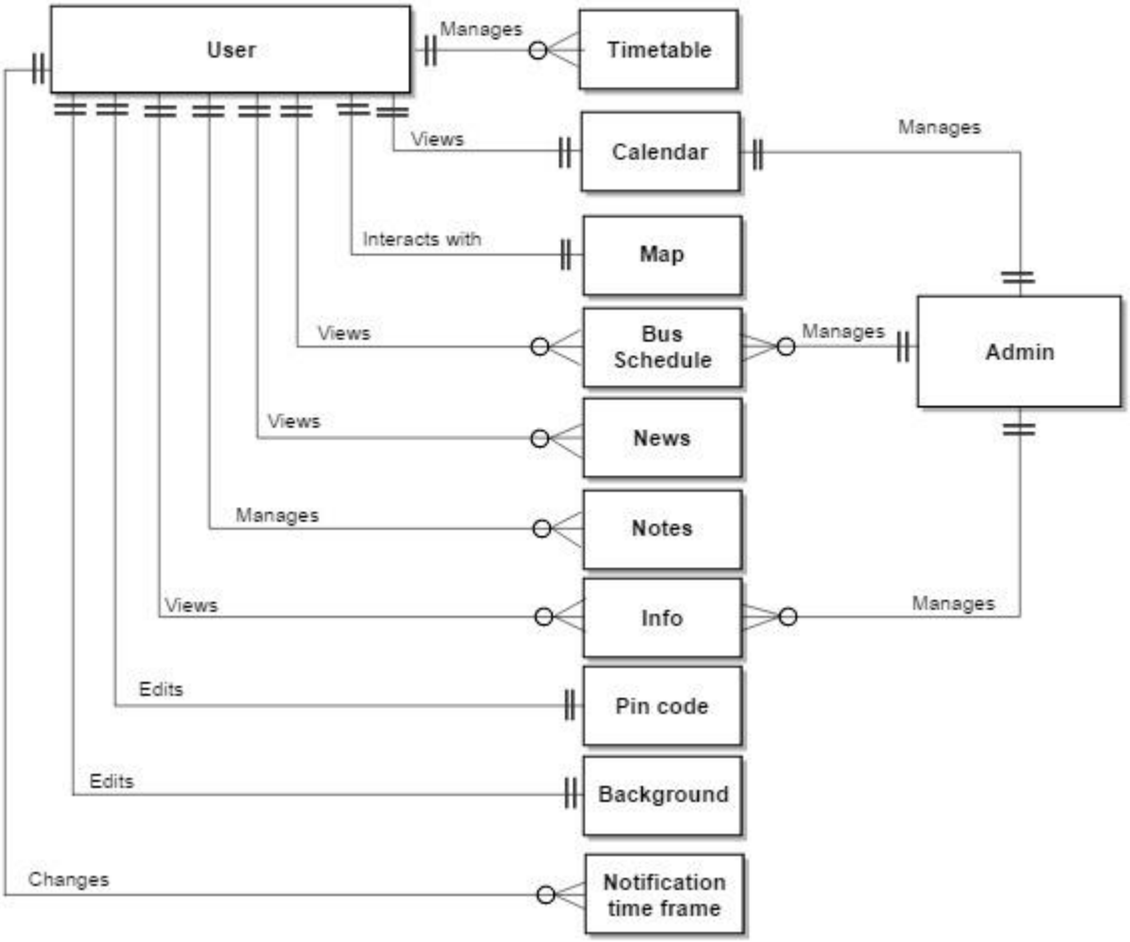


Fig 6. ERD for EMU Student Kit

5. Interface Description:

Each user interface for EMU Student Kit will be explained along with some screenshots in this Section. These user interfaces were created using the Mockflow Wireframe Tool[5].

5.1. Login:

When the user first runs the application, this step will not be required to access the system. The user may then choose to add a PIN code to restrict access to the system. The Advanced Encryption Standard (AES) will be implemented to securely encrypt the PIN code of the user. This is shown in Figure 7.

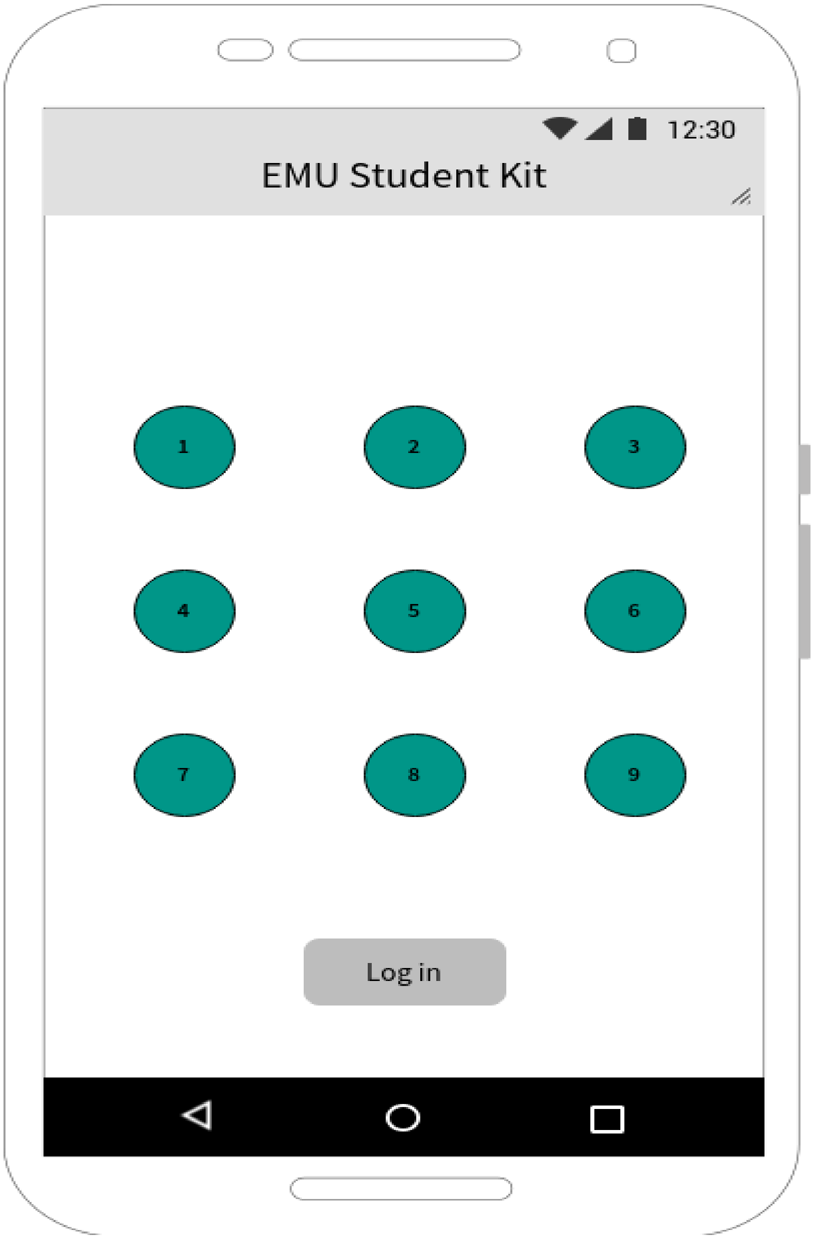


Fig 7. Login Screen

5.2. Main Menu:

This is the main interface where the user can choose what he wishes to do. These functions are shown in Figure 8. Also, the user can click on the "+" sign to perform quick tasks such adding a new note or a new course.

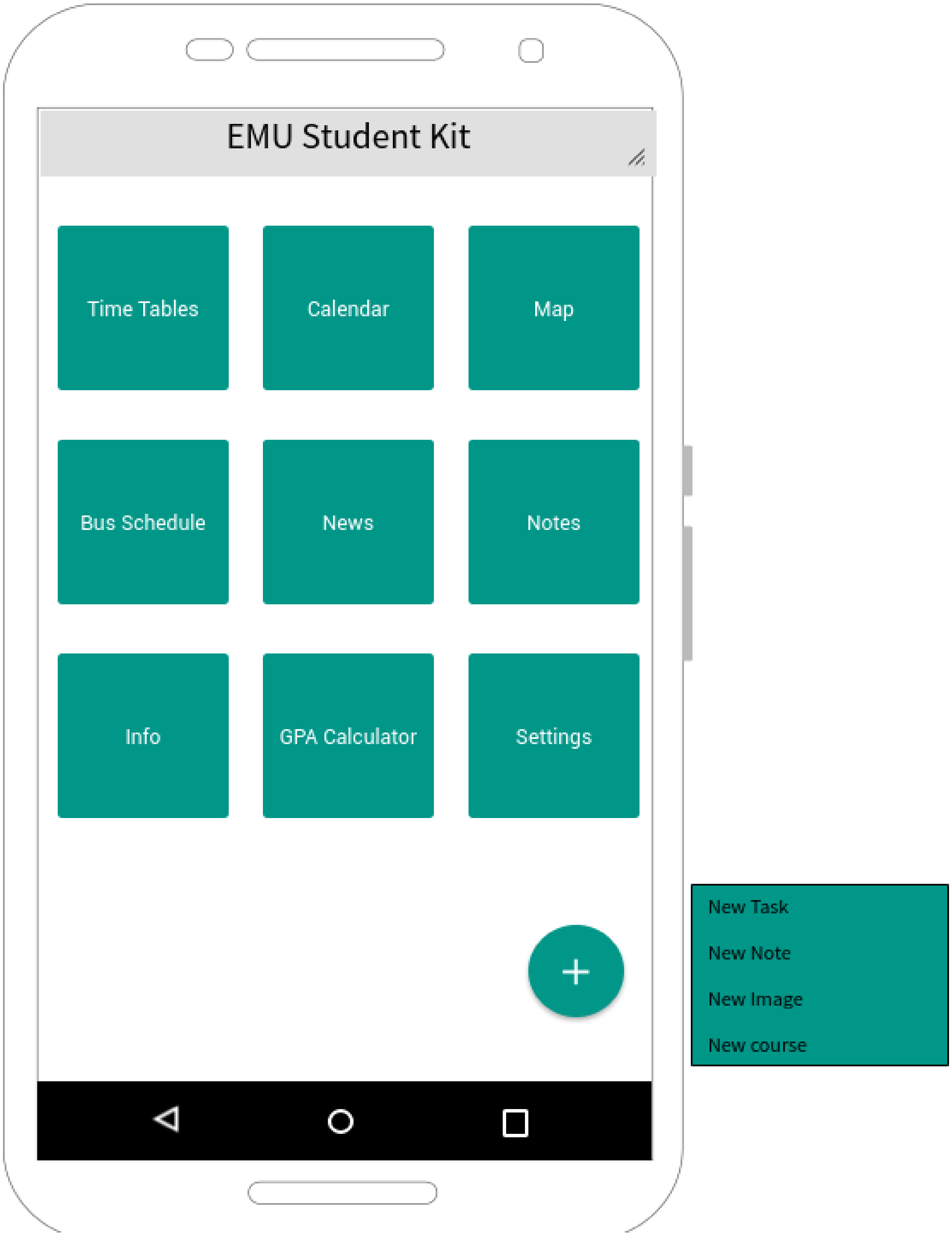


Fig 8. Main Menu

5.3. Time Tables:

In the Time Tables section of the application, there are three main time tables. There is a time table for Courses, Exams and for Tasks. The user can view or edit these time tables. The user can edit a course/exam/task by clicking on it. Then he will be presented with the editing options. This will be shown in Figures 9 to 12.

5.3.1. Time Tables – Courses:

The default menu that is presented when clicking on the Time Tables button.

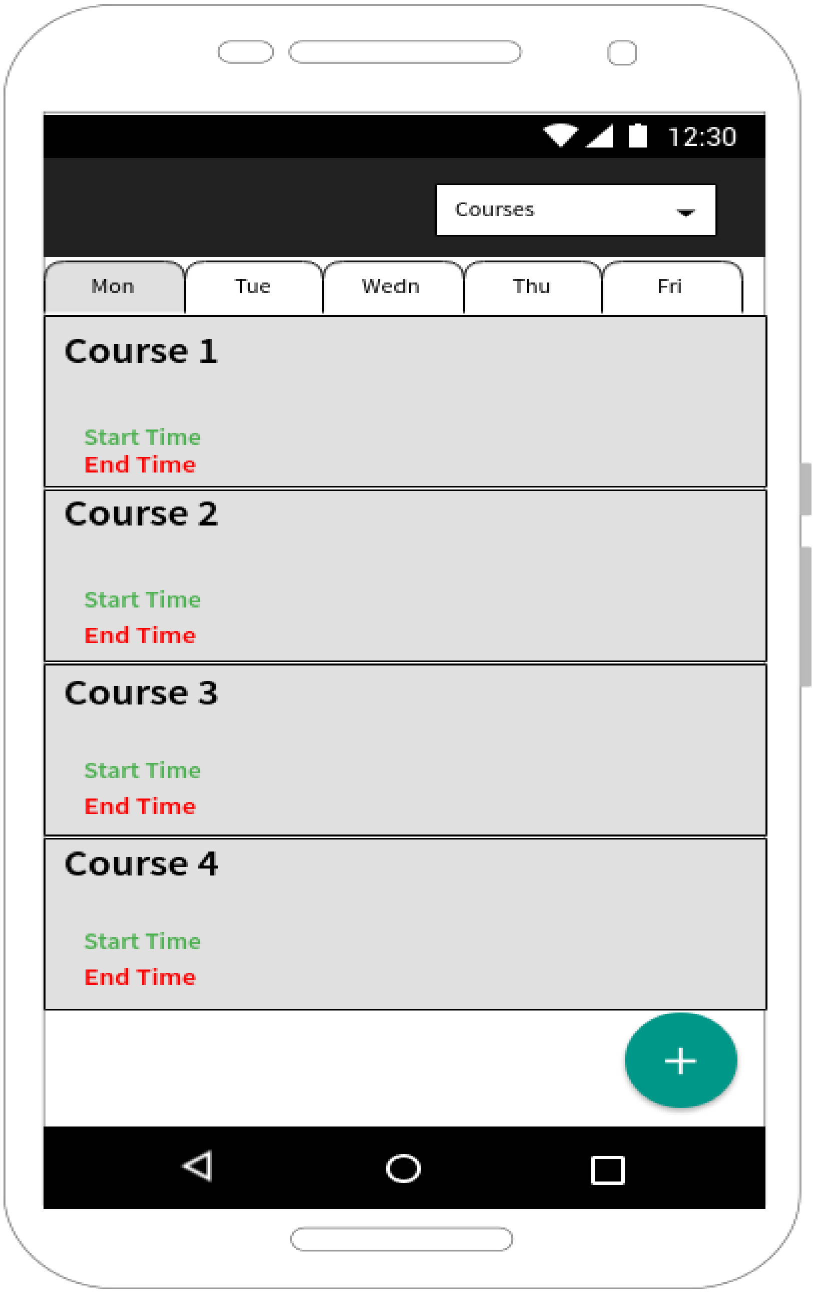
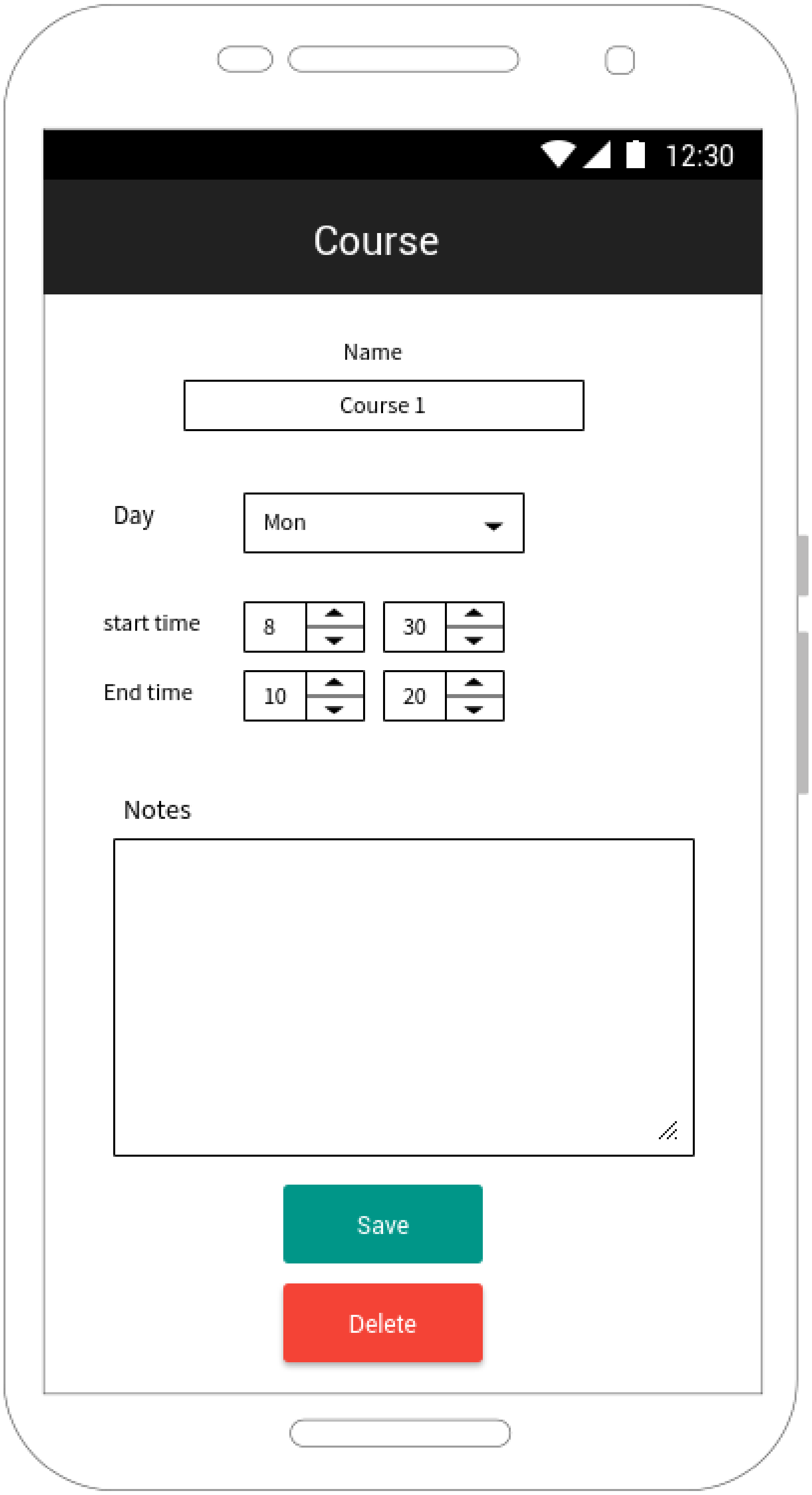


Fig 9. Courses section of the Time Tables

5.3.2. Time Tables – Editing Courses:

Clicking on Course 1 for example will open an editing menu like this. There is a similar menu for the Exams and Tasks section.



Name: Course 1

Day: Mon

start time: 8 30

End time: 10 20

Notes

Fig 10. Editing Course 1 from the list of courses created by the user

5.3.2. Time Tables – Exams:

The user can choose to view the exams time table by clicking the Exams button from the drop down list.

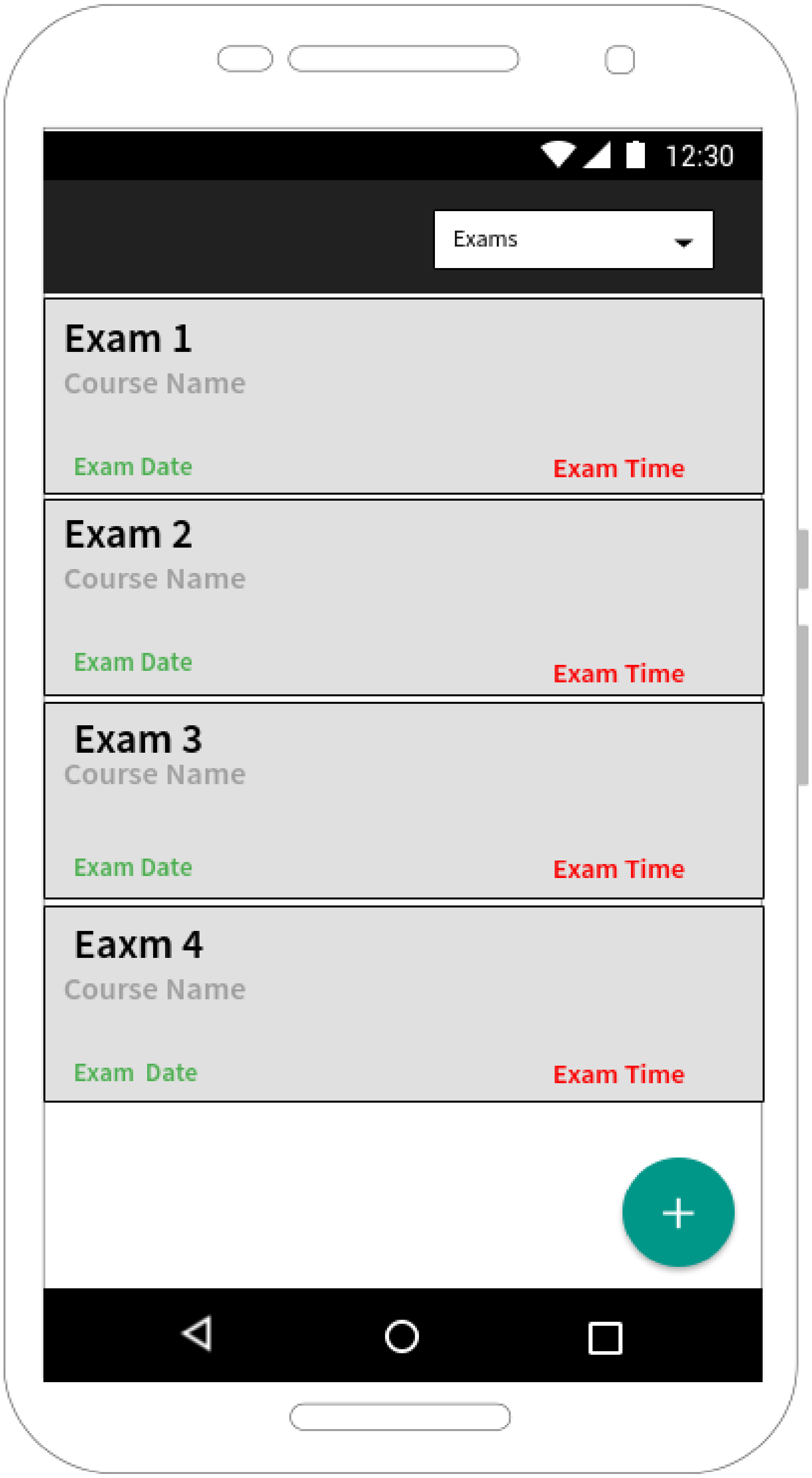


Fig11. Exams section of the Time Tables

5.3.2. Time Tables – Tasks:

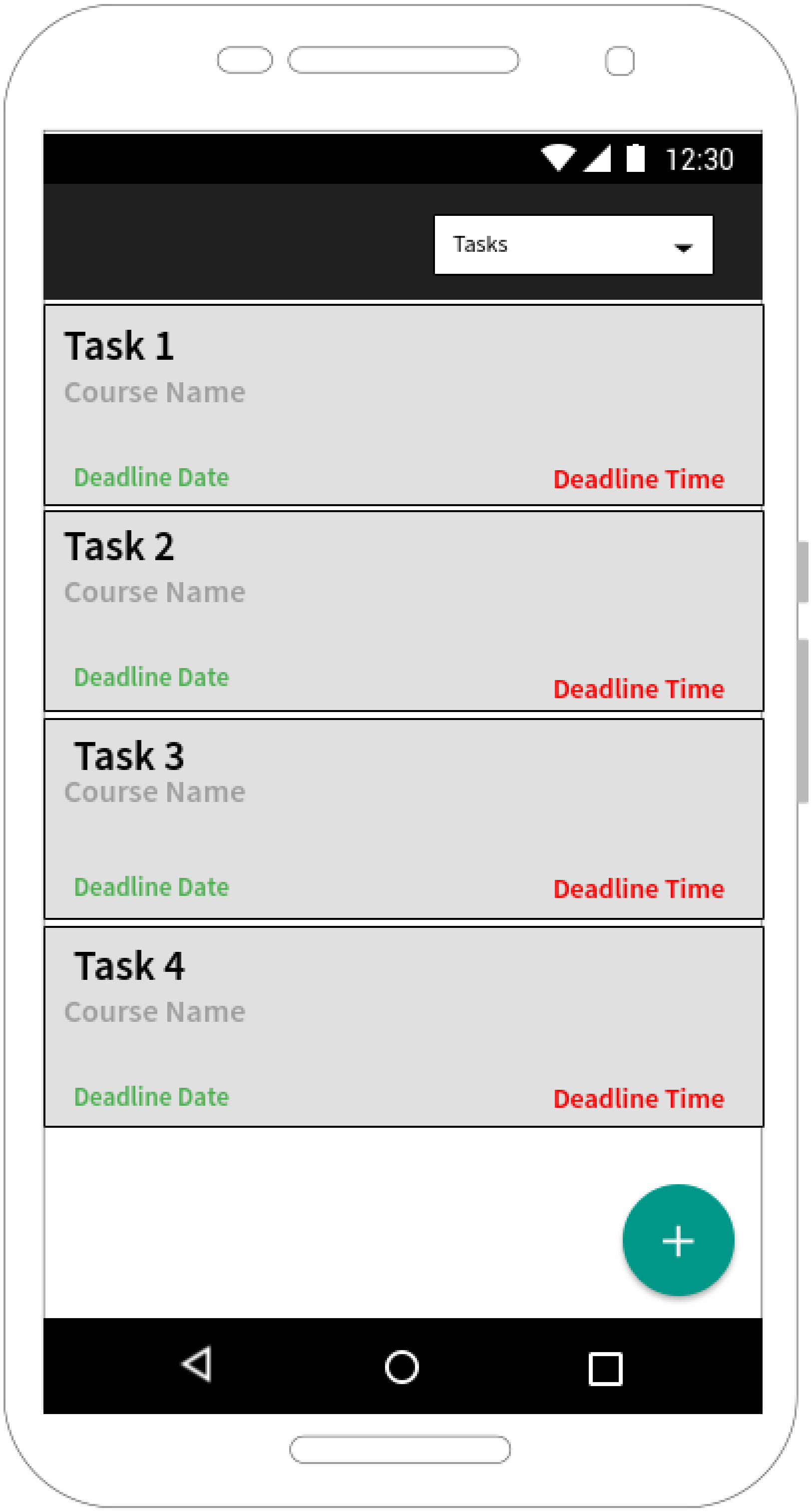


Fig 12. Tasks section of the Time Tables

5.4. Bus Schedules:

This shows all routes, timings and operating days of the buses of EMU. This is shown in Figure 13.

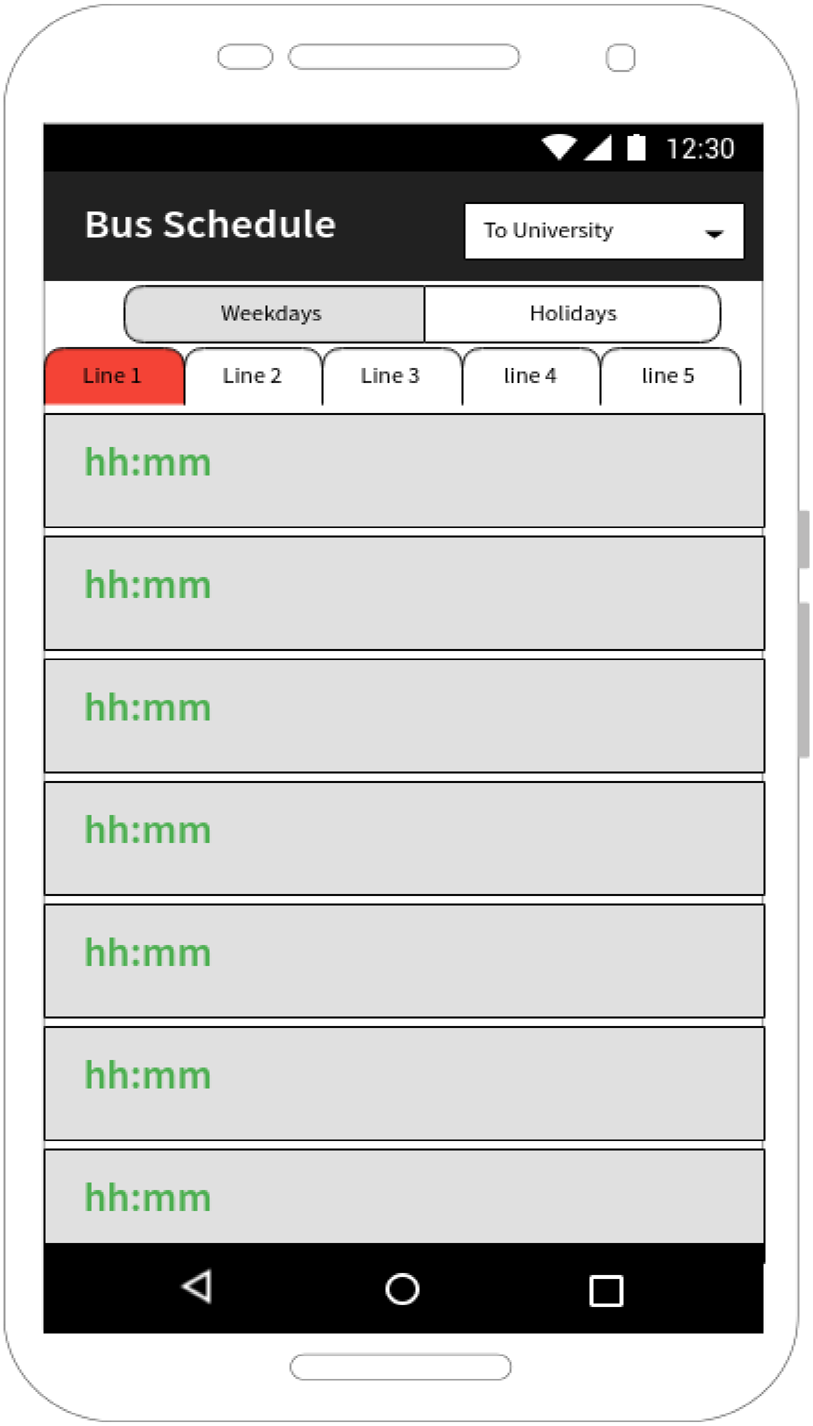


Fig 13. Bus Schedules Interface

5.5. News:

This shows the various News of EMU and they are updated frequently.

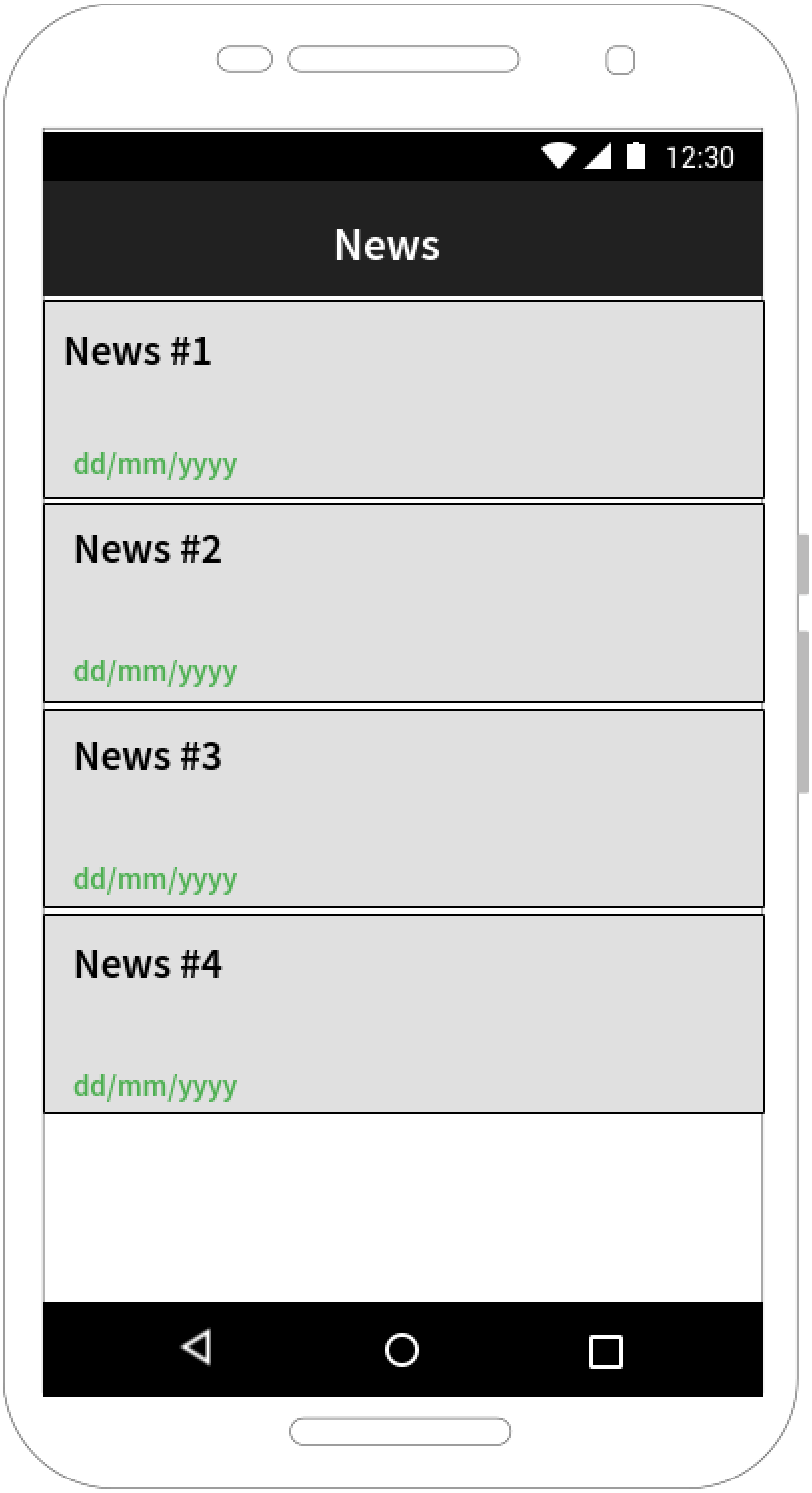


Fig 14. News Interface

5.6. Notes:

The user has the ability to add various text or picture notes and they can be organized for each course. This is shown in Figure 15.

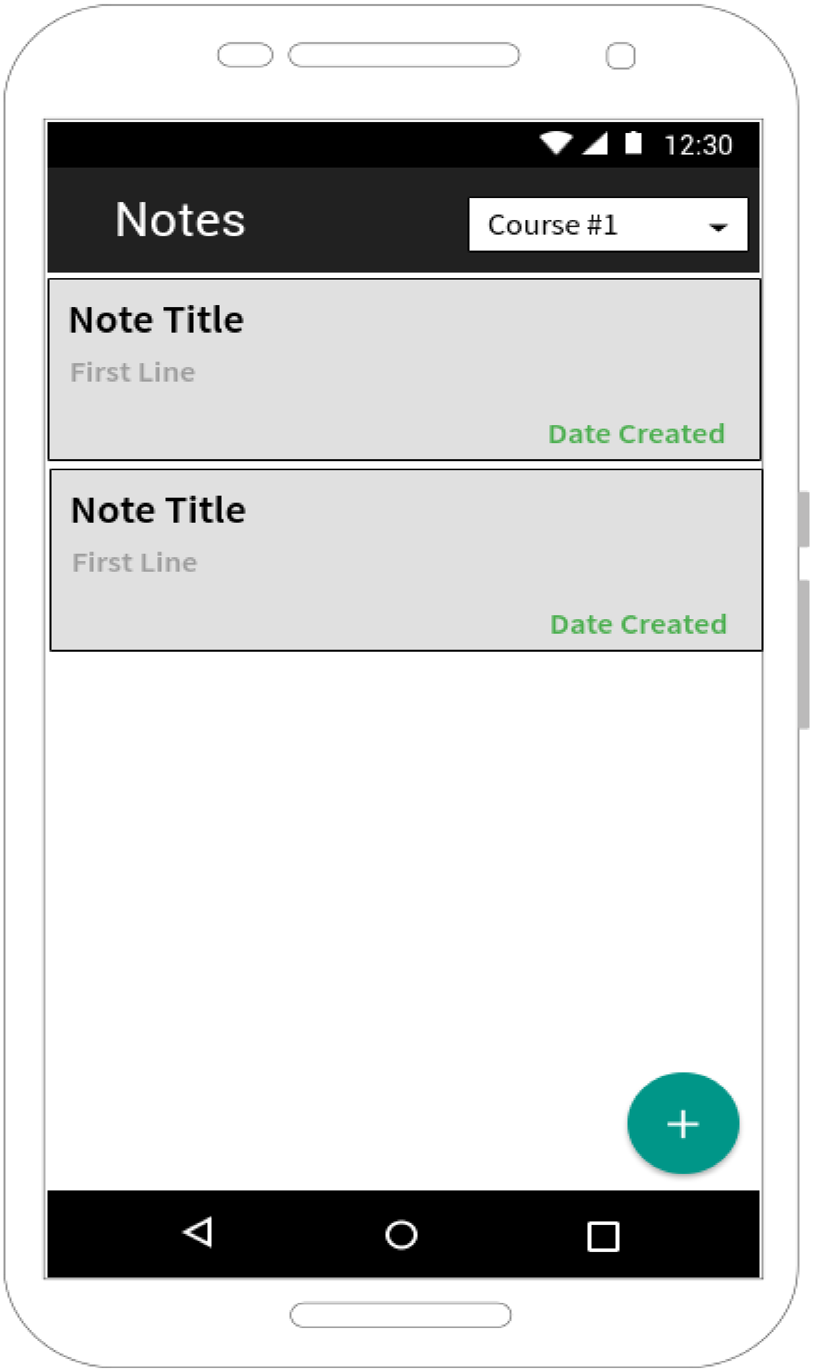


Fig 15. Notes Interface

5.7. Settings:

We can change many of the application’s settings such as:
General: a. add/change PIN. b. change colour/background. c. sync with google drive [yes/no].
d. Notifications: choose time frame [10/15/30 mins]. Notify me (i) before a lecture [yes/no].
(ii) before an event [yes/no].
(iii) before an exam [yes/no].
Help: Various documentations to help the user.
About: Some application related information such as version, date and contact info.

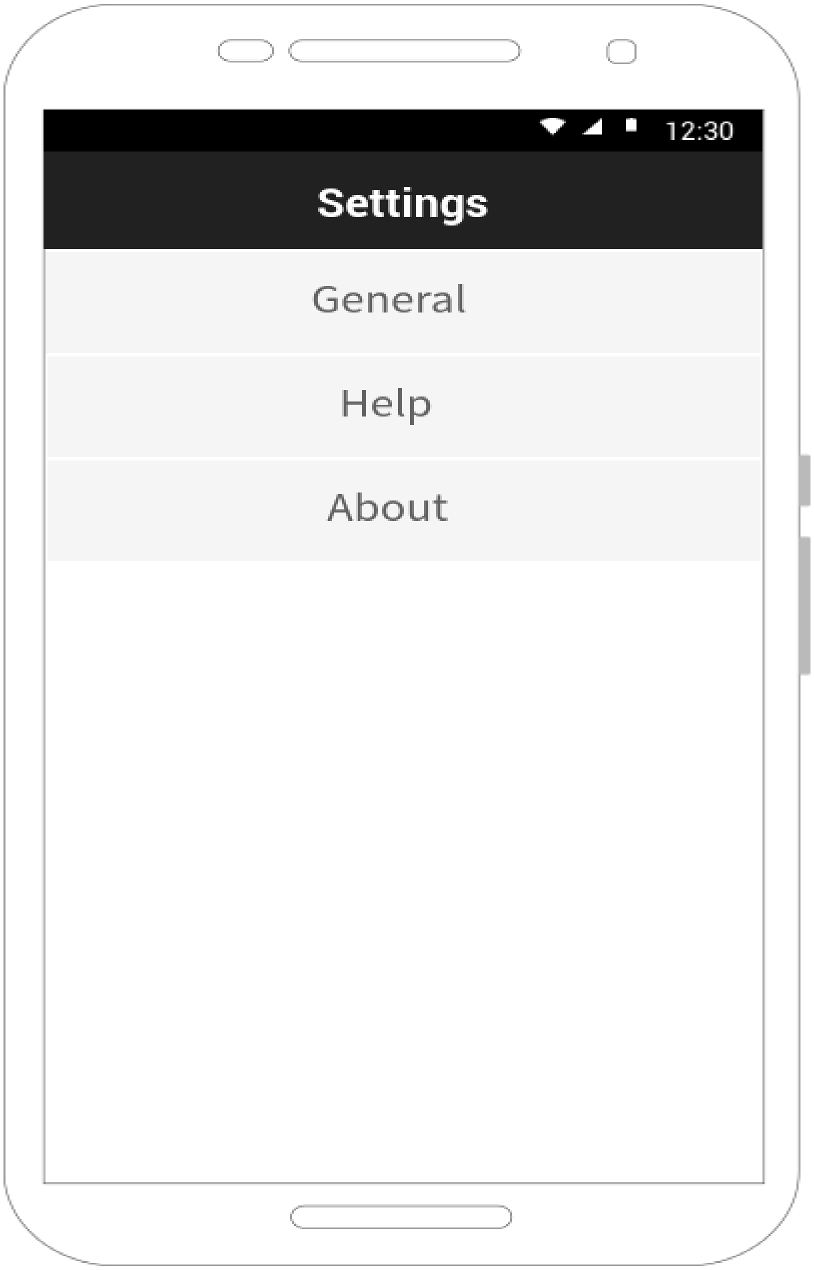


Fig 16. Settings page

5.8. Calendar:

This section shows the university calendar with all the various events and holidays of the year.

5.9. Map:

This section opens an interactive map for EMU campus. All the important places on campus are depicted on this map. When a user clicks on a certain place, information and pictures of that place will be displayed which is very useful for the user. A user can also quickly search for a place he wants to go to and that place will be displayed on the map.

5.10. Info:

This is a quick information section for EMU Students. It can contain many useful information types such as Office hours of various administrative divisions, phone numbers for emergencies, departments, dormitories, etc.

5.11. GPA Calculator:

The GPA/CGPA calculator is a useful and frequently used feature for the students.

6. Detailed Design:

6.1. Module Detailed Design:

Some of the important modules are described in this Section. Also, a Business Process Model which was created using the Microsoft Visio[3] Tool will be shown in Figure 17.

6.1.1. MainMe:
This module runs when we open the program and it organizes and controls what happens whenever we interact with something in the application.
sizes: size of the boxes on main menu
logged: true if PIN is correct
#oncreate()/#onpause()/etc: automatically created functions of the module that manages what happens when clicking something.
graphicmod(): graphical modification of the elements
clickb: click function called when button is clicked
isnetworkavailable: checks if there is an internet connection

6.1.2. Main_Provider:
provide_items: this is a function that retrieves the data from the local DB into the modules.

6.1.3. DB_Constructor:
construct_DB(): When the program is run, it checks if a local database if available. If it is not available, it creates a new local database.

6.1.4. Settingsact: (settings activity)
mclient: azure DB client (may be removed)
mtodotable: azure DB client (may be removed)
pass: function to add/change pin

6.1.5. Un_Map: (university interactive map)
allplaces: list containing all the places on the map
filename: name of the json file containing all the places details
mmap: google map API (application program interface)
onmapready: function of the module, automatically created by google maps

6.1.6. Places_click:
when clicking a place on the map, this entity manages that clicking and what is showed when a user clicks on a place.

6.1.7. Main_Updater:
update(): updates the local DB from the online database managed by the admin

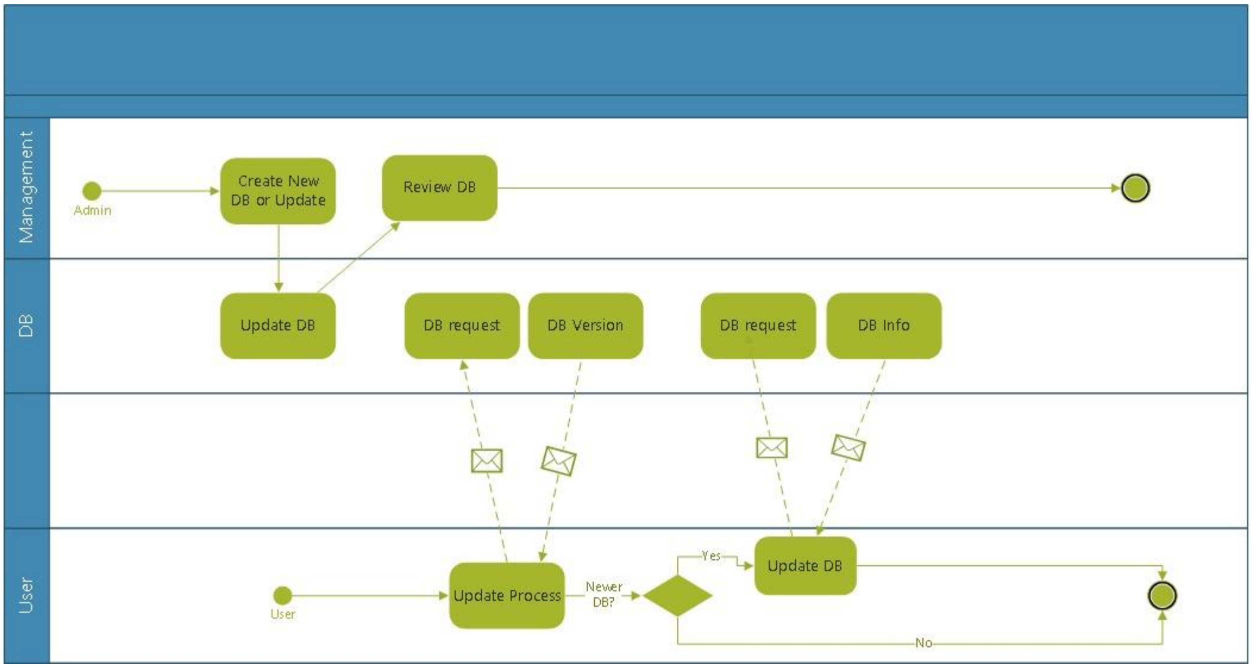


Fig 17. Business Process Model

6.2. Data Detailed Design:

Some important data modules are described in this Section.

6.2.1. TimeTable_Classes:
ID: variable to depict if it is course/exam/task
Course: string to store course name
starting_time: starting time of the course
finishing_time: finishing time of the course
Notes: string of notes the user may take
TimeTable_Classes(): automatically run function of the module
getcourse(): function that retrieves the course name
getDay_of_week(): function that retrieves the day of week
getNotes(): function that retrieves the notes
setCourse(): function that sets the course name for the user
setDay_of_week(): function that sets the day of week of the course for the user
setNotes(): function that sets the notes of the user

6.2.2. Bus_Data:
Line_Number: integer to store bus line number
Weekday: Boolean variable to depict if it is a weekday or a holiday
To_from: Boolean var to determine if the route is going to EMU or coming from EMU
Time: time variable
getLine_Number(): function that gets the line number

getWeekday (): function that gets the weekday
getTo_from (): function that gets the to/from information
getTime (): function that gets the time

6.3. UML Interaction Diagrams:

Two activity diagrams are shown in Figures 18 and 19. These were created using the GenMyModel[6] tool by referring to some ideas from creately.com[7] [8] and slideshare.com[9] websites. The use case diagram shown in Figure 20 was also created using the GenMyModel[6] tool. Also, the two sequence diagrams shown in Figures 21 and 22 were created using the Gliffy[4] tool.

6.3.1. Activity Diagram – Login:

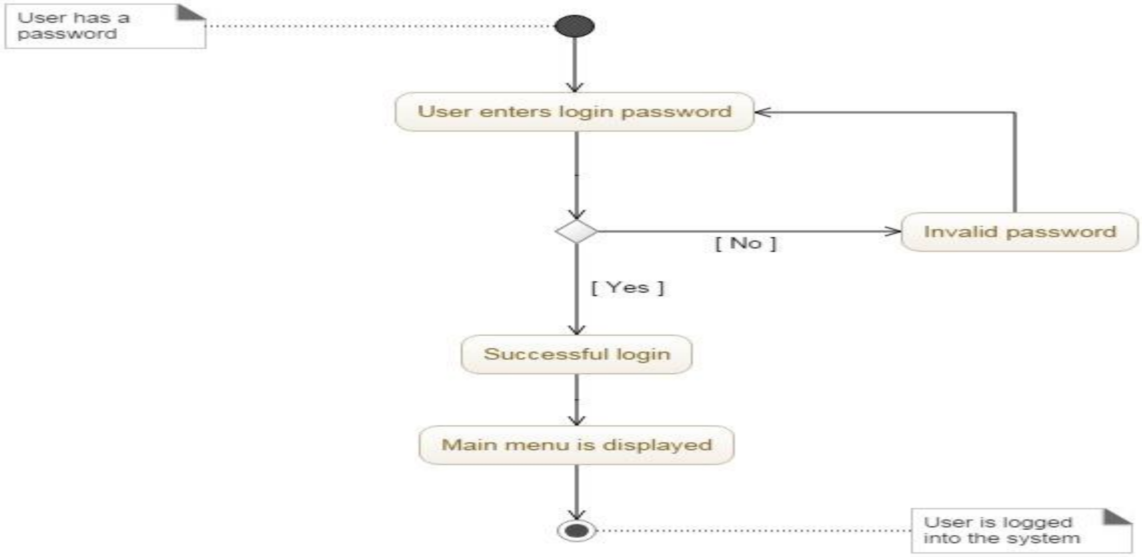


Fig 18. UML Activity Diagram for User Login to System

6.3.2. Activity Diagram – Manage Tasks:

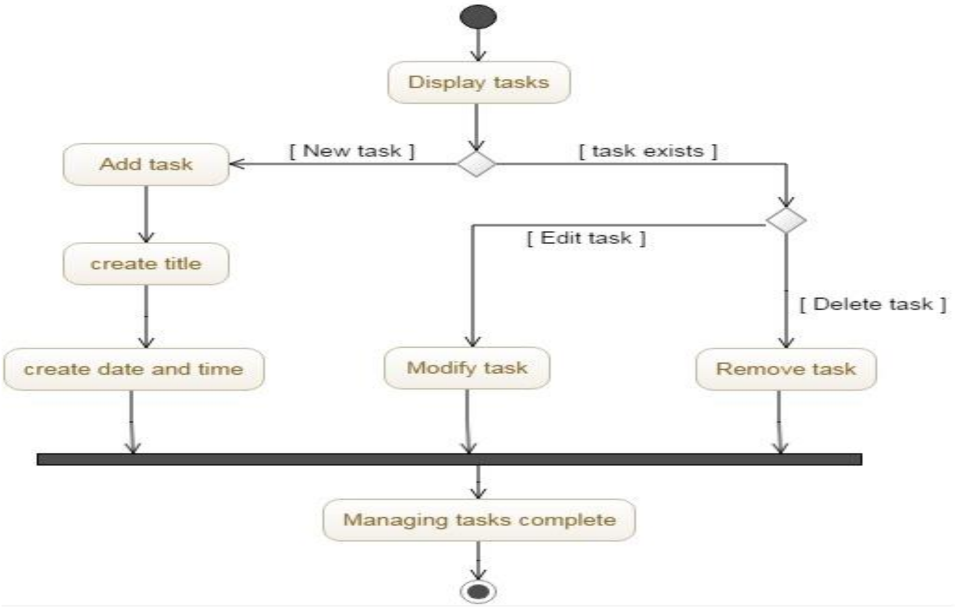


Fig 19. UML Activity Diagram for Managing Tasks

6.3.3. Use Case Diagram:

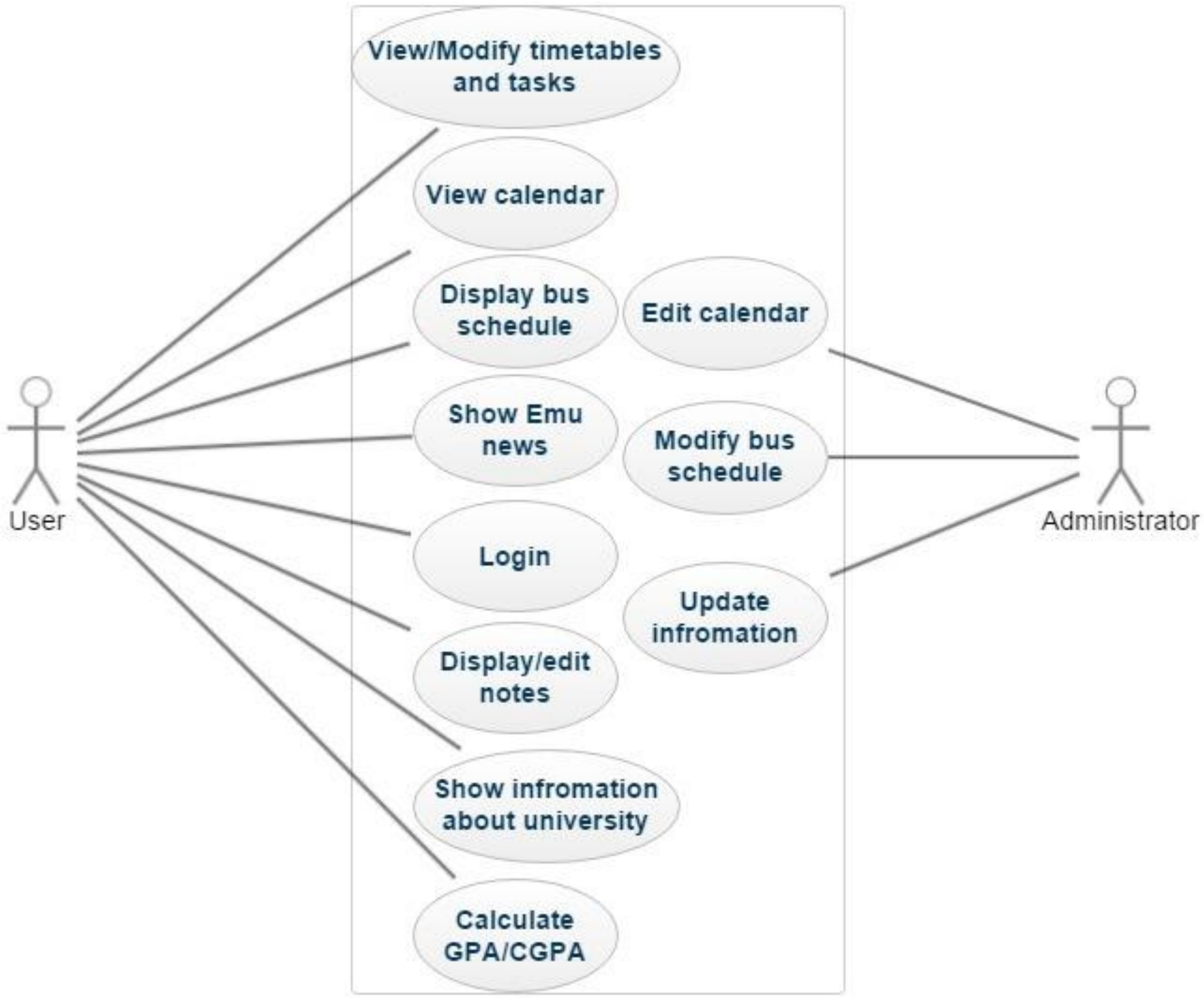


Fig 20. Use Case Diagram

6.3.4. Sequence Diagram - Login:

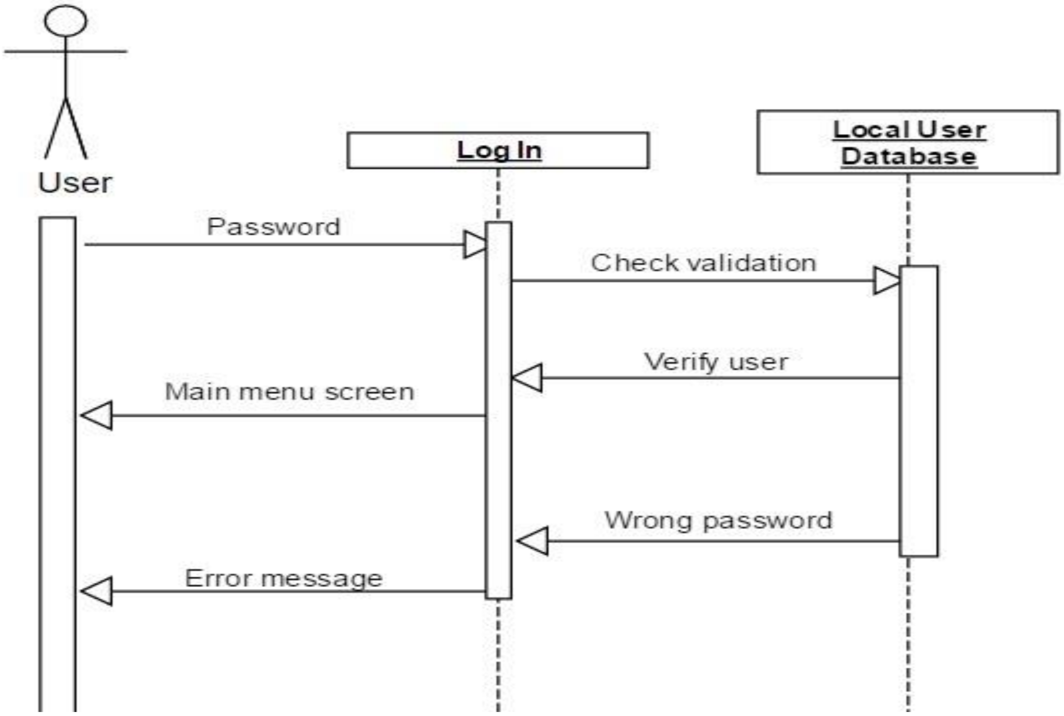


Fig 21. UML Sequence Diagram for User Login to System

6.1.5. Sequence Diagram – Add New Course:

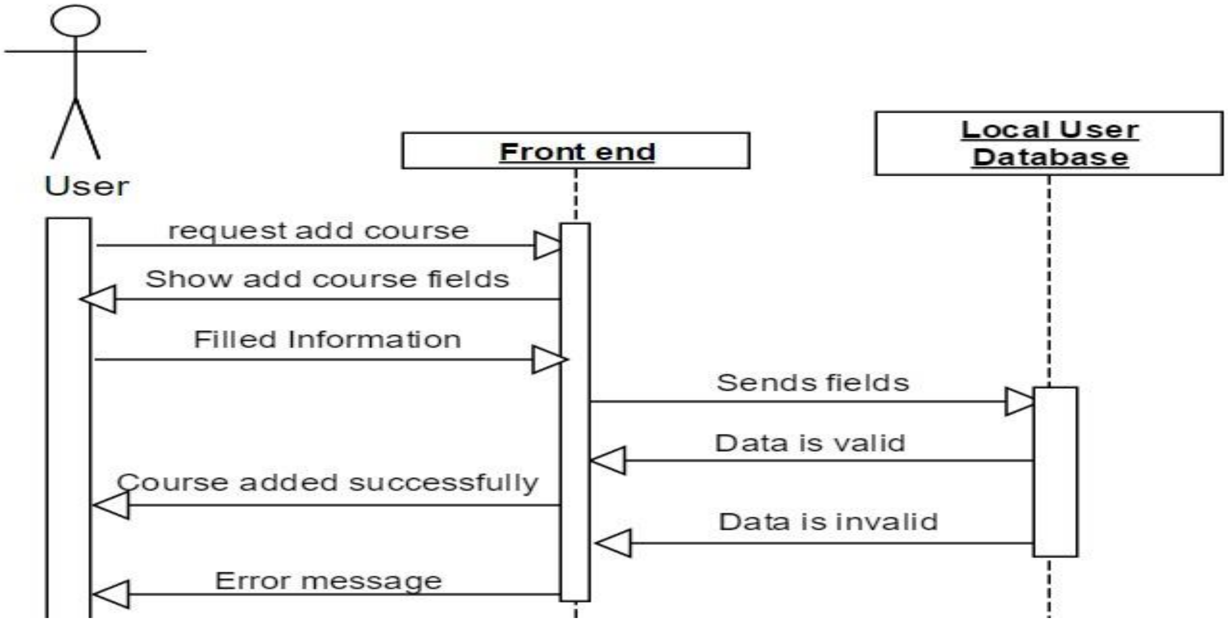


Fig 22. UML Sequence Diagram to Add a new Course