EMU STUDENT KIT

Final Project Report

CMSE 322

Team members: Talal Mahdy - 147139

Adham Moshasha - 148387

Mohamed M. M. Balto - 147697

Abdoulgwad Elsheredi -147597

Computer Engineering Department

Eastern Mediterranean University

Spring 2016-2017

ABSTRACT

This project aims to produce an educational Android Application for EMU Students. The application's objectives are to properly manage the time of the students, increase their performance, organization and productivity. The application will assist students in their lectures, provide them with easily accessible information about EMU, and provide the students with many other different beneficiary functions and features. This project was started due to the shortage of similar integrated educational applications for universities on Google's Play Store. This will give EMU Students an advantage over other universities and will maintain EMU's position as the best University in Cyprus. The application will be developed using the Java programming language with a local database and an administrator controlled online database providing updated data to the system. In the end, the final output of the project is a well designed, well documented, easily accessible and easy to use application called the EMU Student Kit that will also be available on Google's Play Store.

Keywords: EMU Student Kit, educational app, Android, Student, Administrator

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In this project, we should develop a fully functional open source and free to use Android educational Application for EMU called the EMU Student Kit in a period of around 4 months. To do this, we used the Incremental Model of development which is a flexible model for small-medium projects. The main benefiters from this project are the students of EMU. While this application can be used by other students as well, most of the features of the application are intended for EMU Students. The need for this project arises from the fact that in today's world, people are becoming more and more dependent on devices such as smartphones to organize their lives. Also, many business, organizations, and Universities are competing between each other in various fields. One of the reasons that there is an important need for this project is so that EMU maintains a competitive edge over others universities. Before writing this report, two main reports were formed, the first one is the Software Requirements Specification Document[1] (SRS) and the second one is the Software Design Specification Document[2] (SDS). In this report, we will go through all the phases that lead to the building of this project. First of all, the specific requirements of the system will be described. Then, the design of various parts of the system will be shown. After that, the implementation of the system will be explained. Finally, the testing phase will be explained and a user guide will also be shown.

2. PROJECT PLANNING AND MANAGEMENT

2.1. Project Team

Table 1. Project Team

Project No	2
Project Name	EMU Students Kit
Start Date	15-Feb-2017
End Date	12-Jun-2017
Time	

Project Manager/System Architect/Programmer			
Name Surname	Talal Mahdy	**ID No**	147139
Address	Famagusta, North Cyprus		
Phone	+90 533 8885729	**Fax**	
Email	talal.mahdy96@gmail.com		

Lead Programmer/Database Developer/Administrator			
Name Surname	Mohamed M. M. Balto	**ID No**	147697
Address	Famagusta, North Cyprus		
Phone	+90 533 8397554	**Fax**	
Email	Baltu.libya@gmail.com		

Designer/Software Tester/Maintainer			
Name Surname	Abdoulgwad Hussien Elsheredi	**ID No**	147597
Address	Famagusta, North Cyprus		
Phone	+90 533 8528065	**Fax**	
Email	abdoulgwad.elsheredi@yahoo.it		

Requirements Engineer/User Interface Designer			
Name Surname	Adham Moshasha	**ID No**	148387
Address	Famagusta, North Cyprus		
Phone	+90 533 8725650	**Fax**	
Email	adhamoshasha@gmail.com		

2.2. Organization Scheme

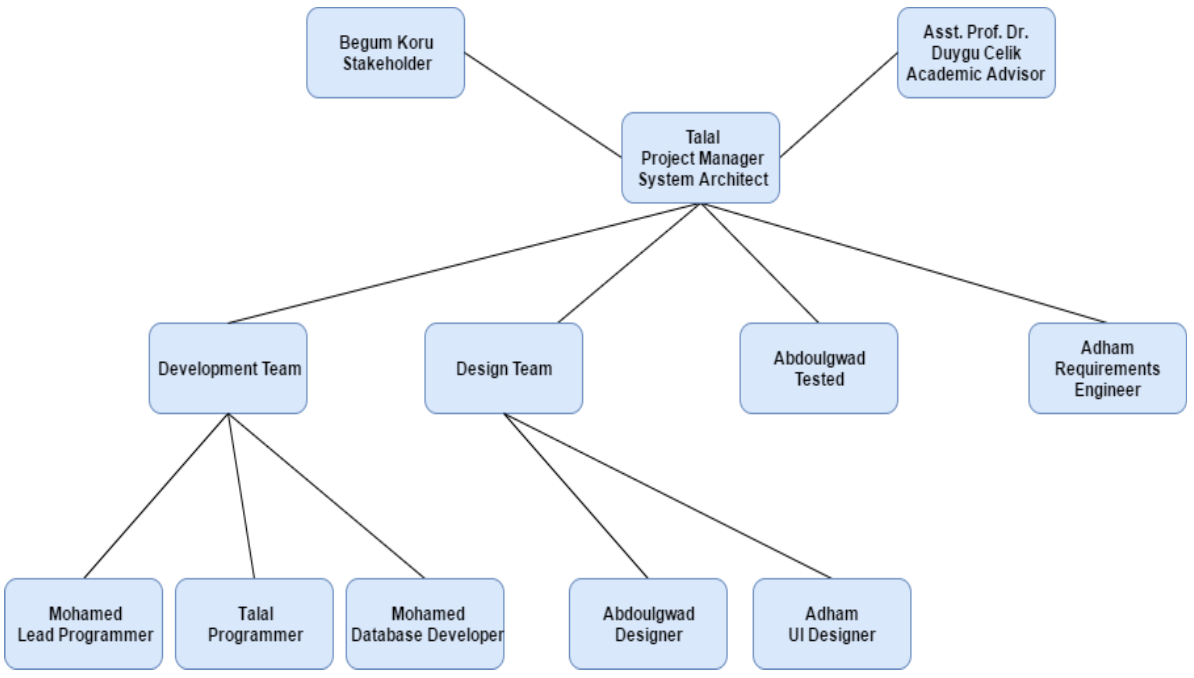


Figure 1. Organization Scheme

2.3. Tools/Methods Applied

Project Management and Scheduling: Microsoft Project[3]

Design: Visual Paradigm[4], Microsoft Visio[5], Mockflow Wireframe[6], Gliffy[7], GenMyModel[8] draw.io[9]

Implementation: Java Programming language, Microsoft SQL Server Management Studio[10], Android Studio IDE[11], Microsoft Azure[12], SQLite[13], JSON[14], Notepad++[15], DB Browser for SQLite[16]

2.4. Reason for starting the Project

These days, people are becoming more and more dependent on devices such as smartphones to organize their lives. There are around 1.2 Billion smartphones around the world. In the US for example, 25% of the population access the internet primarily from their smartphones. These are big figures so our team decided to start a mobile application development strategy to maintain a competitive edge over other educational applications and organizations. It is very important for business organizations to have a responsive website and a mobile application but many businesses have not yet implemented this strategy. For example, in 2014, 45% of all US business organizations did not have a responsive website or a mobile application. We also started this project because students would always want to find what they need quickly. Also, students would always want to stay up to date with all the news that are important to them and get notified immediately. Furthermore, students and other customers almost spend more time using an easy to access app other than a website or a desktop application. Looking at Google's Play Store, there are not many educational applications like EMU Student Kit and it is feasible and makes sense to develop a project like this.

2.5. Success Criteria

Success Criteria are metrics to determine if the project is successful. Some of them are:

a. Total Downloads: It is estimated that at least 80% of EMU students having Android devices will download the app. But this app can also be used by other students.
b. Monthly Average Users (MAU: The application should have a high number of Average users among those who downloaded the application. If it appears that the MAU is growing, then the project is growing in the right path.
c. Engagement: Also, those users should have a high engagement ratio, i.e., users visit it frequently and use it for a considerable amount of time. Engagement can be measured by metrics such as session length (time period between app open and close), session interval (time between the user's first session and their next one) and retention rate (users who return to your app based on the date of their first visit).
d. Documentation: The number of users submitting questions or help requests should be less due to well documentation and user guide.

2.6. Software Development Plan

In this project, we will be applying an evolutionary development approach. An evolutionary development is based on the idea of developing an initial implementation, exposing it to the customer's comments, refining it through many versions until an adequate system has been developed. This development method is going to be more effective in this project since it is a small-medium sized system. The requirements for this system are not well defined from the beginning and we have to work with the customer and produce prototypes while obtaining feedbacks from the customer. To do this, we are going to have to conduct a number of interviews with the customer and clearly understand the requirements. The main advantages of using this approach to develop this project are:

- Each module passes through requirements, design, implementation and testing phases.
- Most important Modules are developed first.
- It is more flexible and less costly to change requirements.
- It is easier to test and debug the modules.
- Delivery of initial Modules is quick.

2.7. Work Packages and Gantt Chart

Table 2. Work Package 1

Work Package No	1
Work Package Name	**Feasibility and Pre-Research (SRS stage)**
Start-End Date and Time	Start: 20-02-17 Finish: 09/03/17

1- Activities of work package.
1. Scope. **2. Analysis/Software Requirements.**
2- Methods and parameters that will be used for work package.
None.
3- Experiments, tests and analysis in the work package.
1. Scope: 1.1. Determine project scope 1.2. Secure project approval 1.3. Define preliminary resources 1.4. Secure core resources 1.5. Scope complete **2. Analysis/Software Requirements:** 2.1. Conduct needs analysis 2.2. Draft preliminary software specifications 2.3. Develop preliminary budget 2.4. Review software specifications/budget with team 2.5. Incorporate feedback on software specifications 2.6. Develop delivery timeline 2.7. Obtain approvals to proceed (concept, timeline, budget) 2.8. Secure required resources 2.9. Analysis complete
4- Output of work package and its success criteria.
Outputs: Initial Requirements Specification Document (SRS), feasibility analysis, secured resources. **Success Criteria:** Project approved, project is feasible to implement, initial requirements are well documented, resources and team members are secured.
5- Relation of output with other work packages
This is the initial phase of development and is the basic input for all other work packages. It defines the following: What is the project? Who are the stakeholders? Who will use the system? How should it be developed? Who are the team members? What are the basic requirements? How should it be developed? How should it be delivered?...

Table 3. Work Package 2

Work Package No	2
Work Package Name	**System Design (SDS Stage)**
Start-End Date and Time	Start: 09-03-17 Finish: 31/03/17

1- Activities of work packages.
1. EMU Student Kit Software Design **2. Development of first prototype** **3. Improve SRS Document**
2- Methods and parameters that will be used for work package.
Visual Paradigm, Microsoft Visio, Mockflow Wireframe, Gliffy, GenMyModel, draw.io
3- Experiments, tests and analysis in the work package.
Review preliminary software specifications Develop functional specifications Design of System Develop prototype based on functional specifications Review functional specifications and Design Incorporate feedback into functional specifications Obtain approval to proceed Design complete
4- Output of work package and its success criteria.
Outputs: A Software Design Specification (SDS) Document, First Prototype of Software. **Success Criteria:** An improvement of the SRS Document as a result of better understanding of requirements from first prototype, completion of system design.
5- Relation of output with other work packages
The design stage is the next stage in the software development life cycle. Without designing the software and knowing what has to be done, it will be very difficult for the programmer to develop the software and many mistakes will be done. So this work package is a very important prerequisite to the next stage which is the development stage.

Table 4. Work Package 3

Work Package No	3
Work Package Name	**Software Development Stage**
Start-End Date and Time	Start: 01-04-17 Finish: 23/04/17

1- Activities of work packages.
The main coding, primary debugging of the program and development of the database.
2- Methods and parameters that will be used for work package.
Java Programming language Microsoft SQL Server Management Studio, Android Studio IDE, Microsoft Azure, SQLite, JSON
3- Experiments, tests and analysis in the work package.
Review functional specifications Identify modular/tiered design parameters Assign development staff Develop Code and Database Developer testing (primary debugging) Development complete
4- Output of work package and its success criteria.
Outputs: EMU Student Kit Android Application Package (APK) **Success Criteria:** A successful working APK file of our project.
5- Relation of output with other work packages
During the development of our application, the coders will obviously find some bugs and attempt to fix them. However, there might be some logical or other types of errors that a developer might not notice. Therefore, it is important for the application to be tested by a separate dedicated tester. Testing of the application can begin shortly after the development of the first unit of the application.

Table 5. Work Package 4

Work Package No	4
Work Package Name	**Software Testing Stage**
Start-End Date and Time	Start: 15-04-17 Finish: 23/05/17

1- Activities of work packages.
1. **Unit and Integration Test Plans.** 2. **Unit Testing.** 3. **Integration Testing.**
2- Methods and parameters that will be used for work package.
None
3- Experiments, tests and analysis in the work package.
1. Unit and Integration Test Plans: 1.1. Develop unit test plans using product specifications 1.2. Develop integration test plans using product specifications **2. Unit Testing:** 2.1. Review modular code 2.2. Test component modules to product specifications 2.3. Identify anomalies to product specifications 2.4. Modify code 2.5. Re-test modified code 2.6. Unit testing complete **3. Integration Testing:** 3.1. Test module integration 3.2. Identify anomalies to specifications 3.3. Modify code 3.4. Re-test modified code 3.5. Integration testing complete
4- Output of work package and its success criteria.
Outputs: Test data, verification results **Success Criteria:** Testing successfully completed with all the errors and bugs successfully fixed.
5- Relation of output with other work packages
After successfully testing the system, next stages in the software life cycle are the delivery and maintenance stages. The software should be delivered and installed as per the request of the customer. Also, the maintenance stage is very important as a software may serve for many years to come and it will obviously need to be updated. Therefore, a good maintenance team along with good documentation are very important for the product to be successful.

Table 6. Work Package 5

Work Package No	5
Work Package Name	**Documentation and Delivery**
Start-End Date and Time	Start: 09-03-17 Finish: 03/06/17

1- List the activities of work packages.

1. **Documentation**
2. **Pilot**
3. **Deployment**
4. **Post Implementation Review**

2- Describe the methods and parameters that will be used for work package.

None.

3- List the experiments, tests and analysis in the work package.

1. **Documentation**
   - 1.1. Develop Help specification
   - 1.2. Develop SRS Document
   - 1.3. Develop SDS Document
   - 1.4. Develop Help system
   - 1.5. Review Help documentation
   - 1.6. Incorporate Help documentation feedback
   - 1.7. Develop user manuals specifications
   - 1.8. Develop user manuals
   - 1.9. Review all user documentation
   - 1.10. Incorporate user documentation feedback
   - 1.11. Documentation complete
2. **Pilot**
   - 2.1. Identify test group
   - 2.2. Develop software delivery mechanism
   - 2.3. Install/deploy software to Google's Play Store
   - 2.4. Obtain user feedback
   - 2.5. Evaluate testing information
   - 2.6. Pilot complete
3. **Deployment**
   - 3.1. Determine final deployment strategy
   - 3.2. Develop deployment methodology
   - 3.3. Secure deployment resources
   - 3.4. Train support staff
   - 3.5. Deploy software
   - 3.6. Deployment complete
4. **Post Implementation Review**
   - 4.1. Document lessons learned
   - 4.2. Distribute to team members
   - 4.3. Create software maintenance team
   - 4.4. Post implementation review complete

4- List the output of work package and its success criteria.

Outputs: Successful delivery of project, uploading to Play Store, completed documentation
Success Criteria: A completed well documented, well perceived software application.

5- Explain the relation of output with other work packages

As can be noticed, the documentation stage started at an early time in the software process, sometime after the design stage started. It is important to document all requirements and design aspects of the project along with a proper user guide before delivering the application.

ID	ℹ	Task Mode	Task Name	Duration	Start	Finish
0			**Software Development**	**104 days**	**Mon 20-02-17**	**Sat 03-06-17**
1			**Feasibility and Pre-Research (SRS Stage)**	**17.5 days**	**Mon 20-02-17**	**Thu 09-03-17**
2			**Scope**	**3.5 days**	**Mon 20-02-1**	**Thu 23-02-17**
3			Determine project scope	4 hrs	Mon 20-02-17	Mon 20-02-17
4			Secure project approval	1 day	Mon 20-02-17	Tue 21-02-17
5			Define preliminary resources	1 day	Tue 21-02-17	Wed 22-02-17
6			Secure core resources	1 day	Wed 22-02-17	Thu 23-02-17
7			Scope complete	0 days	Thu 23-02-17	Thu 23-02-17
8			**Analysis/Software Requiremer**	**14 days**	**Thu 23-02-17**	**Thu 09-03-17**
9			Conduct needs analysis	5 days	Thu 23-02-17	Tue 28-02-17
10			Draft preliminary software specifications	3 days	Tue 28-02-17	Fri 03-03-17
11			Develop preliminary budget	2 days	Fri 03-03-17	Sun 05-03-17
12			Review software specifications/budget with team	4 hrs	Sun 05-03-17	Sun 05-03-17
13			Incorporate feedback on software specifications	1 day	Mon 06-03-17	Mon 06-03-17
14			Develop delivery timeline	1 day	Tue 07-03-17	Tue 07-03-17
15			Obtain approvals to proceed (concept, timeline, budget)	4 hrs	Wed 08-03-17	Wed 08-03-17
16			Secure required resources	1 day	Wed 08-03-17	Thu 09-03-17
17			Analysis complete	0 days	Thu 09-03-17	Thu 09-03-17

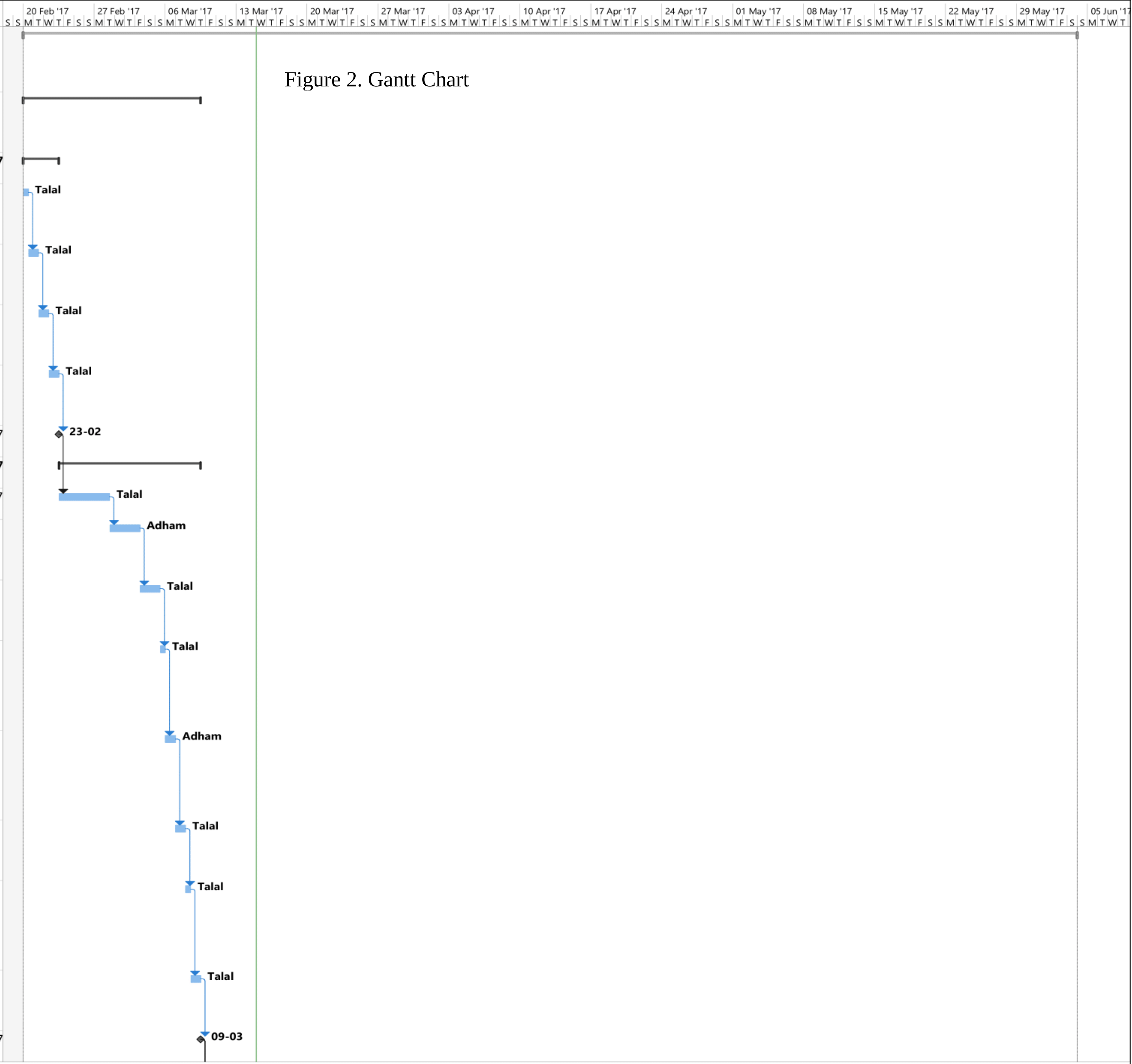
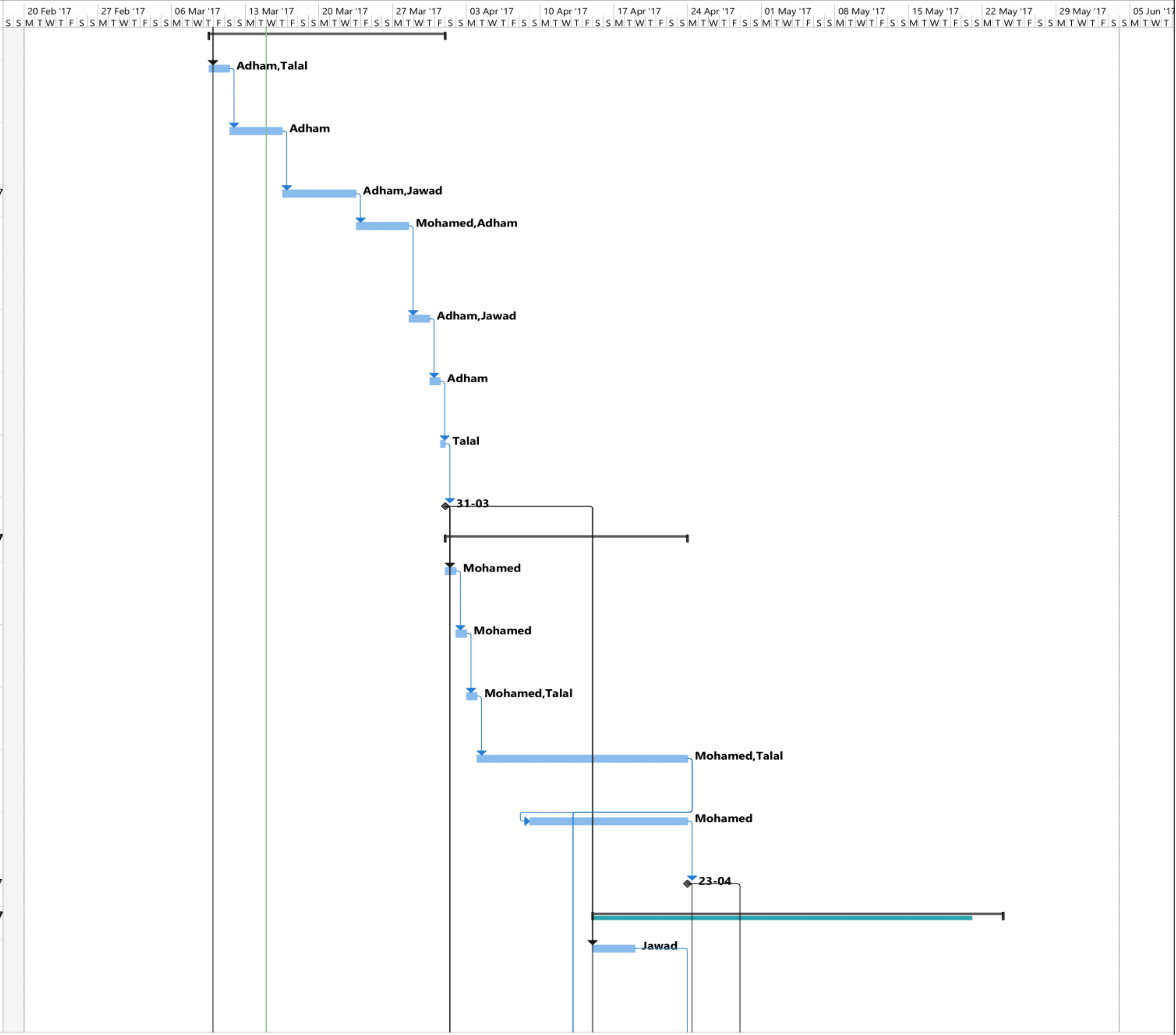


Figure 2. Gantt Chart

Project: Software Development
Date: Wed 15-03-17

Task	▬	Summary	┌─┐	Inactive Milestone	◇	Duration-only	▬	Start-only	[	External Milestone	◆	Manual Progress	▬
Split	┈	Project Summary	┌─┐	Inactive Summary	┌─┐	Manual Summary Rollup	▬	Finish-only	]	Deadline	⬇		
Milestone	◆	Inactive Task	▭	Manual Task	▬	Manual Summary	┌─┐	External Tasks	▬	Progress	▬		

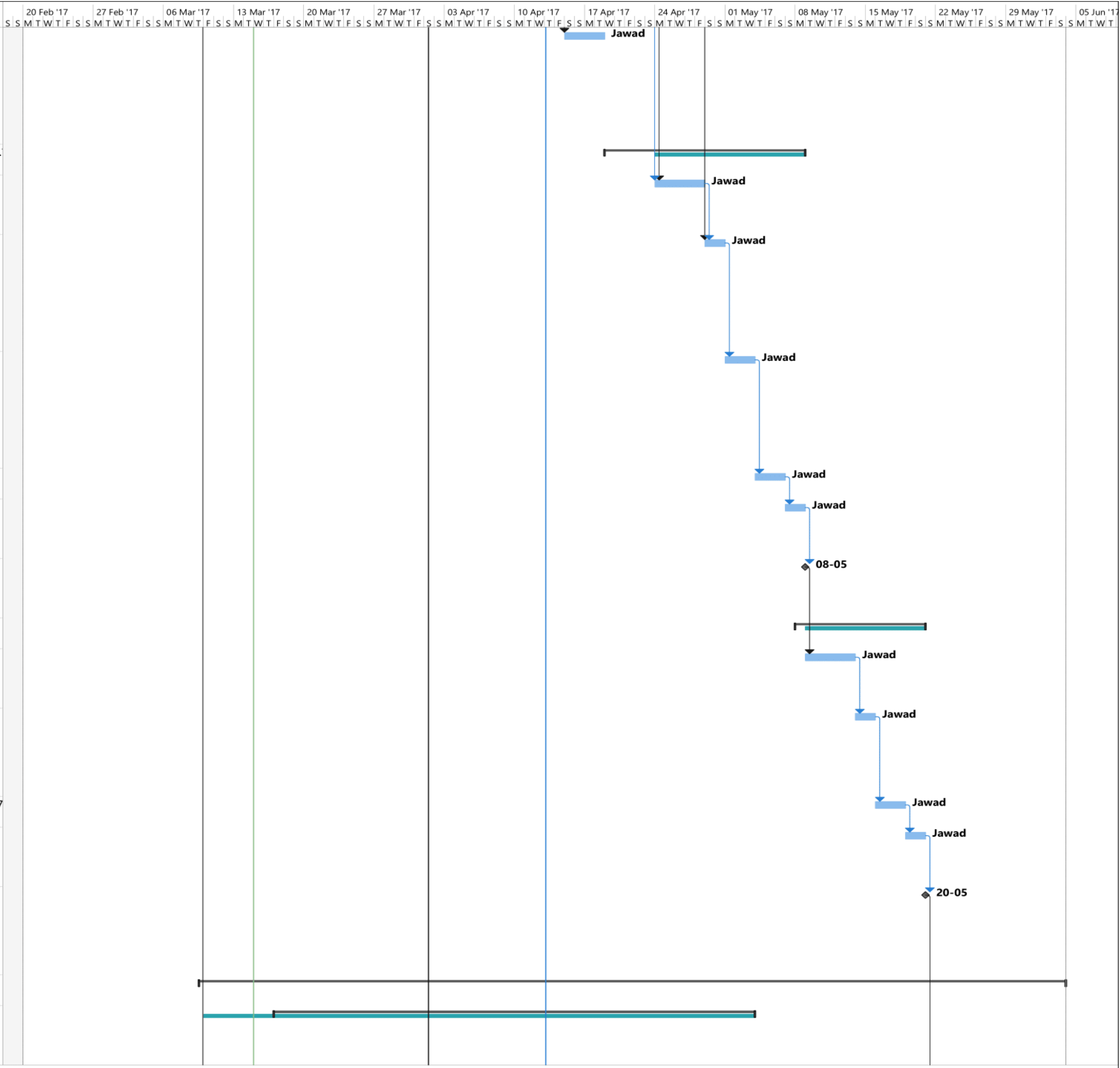
ID		Task Mode	Task Name	Duration	Start	Finish
18			**System Design (SDS Stage)**	**22.5 days**	**Thu 09-03-17**	**Fri 31-03-17**
19			Review preliminary software specifications	2 days	Thu 09-03-17	Sat 11-03-17
20			Develop functional specifications	5 days	Sat 11-03-17	Thu 16-03-17
21			Design of System	7 days	Thu 16-03-17	Thu 23-03-17
22			Develop prototype based on functional specifications	5 days	Thu 23-03-17	Tue 28-03-17
23			Review functional specifications and Design	2 days	Tue 28-03-17	Thu 30-03-17
24			Incorporate feedback into functional specifications	1 day	Thu 30-03-17	Fri 31-03-17
25			Obtain approval to proceed	4 hrs	Fri 31-03-17	Fri 31-03-17
26			Design complete	0 days	Fri 31-03-17	Fri 31-03-17
27			**Software Development Stage**	**23 days**	**Sat 01-04-17**	**Sun 23-04-17**
28			Review functional specifications	1 day	Sat 01-04-17	Sat 01-04-17
29			Identify modular/tiered design parameters	1 day	Sun 02-04-17	Sun 02-04-17
30			Assign development staff	1 day	Mon 03-04-17	Mon 03-04-17
31			Develop Code and Database	20 days	Tue 04-04-17	Sun 23-04-17
32			Developer testing (primary debugging)	15 days	Sun 09-04-17	Sun 23-04-17
33			Development complete	0 days	Sun 23-04-17	Sun 23-04-17
34			**Software Testing Stage**	**39 days**	**Sat 15-04-17**	**Tue 23-05-17**
35			Develop unit test plans using product specifications	4 days	Sat 15-04-17	Tue 18-04-17



Project: Software Development
Date: Wed 15-03-17

Task	▬	Summary	┌─┐	Inactive Milestone	◇	Duration-only	▬	Start-only	[	External Milestone	◆	Manual Progress	─
Split	┈┈	Project Summary	┌─┐	Inactive Summary	┌─┐	Manual Summary Rollup	▬	Finish-only	]	Deadline	⬇		
Milestone	◆	Inactive Task	▭	Manual Task	▬	Manual Summary	┌─┐	External Tasks	▬	Progress	─		

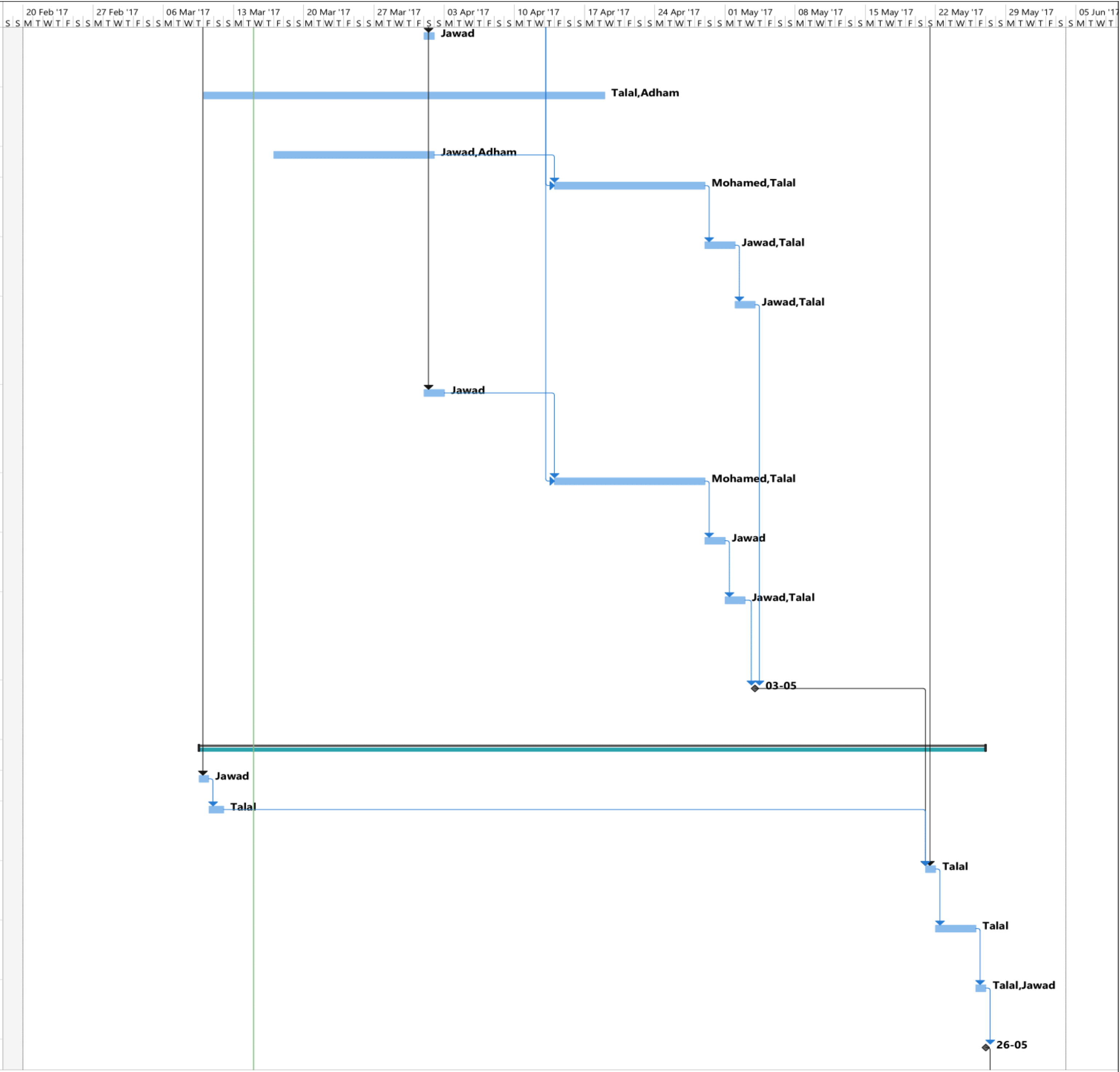
ID		Task Mode	Task Name	Duration	Start	Finish
36			Develop integration test plans using product specifications	4 days	Sat 15-04-17	Tue 18-04-17
37			**Unit Testing**	**20 days**	**Wed 19-04-1**	**Mon 08-05-1**
38			Review modular code	5 days	Mon 24-04-17	Fri 28-04-17
39			Test component modules to product specifications	2 days	Sat 29-04-17	Sun 30-04-17
40			Identify anomalies to product specifications	3 days	Mon 01-05-17	Wed 03-05-17
41			Modify code	3 days	Thu 04-05-17	Sat 06-05-17
42			Re-test modified code	2 days	Sun 07-05-17	Mon 08-05-17
43			Unit testing complete	0 days	Mon 08-05-17	Mon 08-05-17
44			**Integration Testing**	**13 days**	**Mon 08-05-1**	**Sat 20-05-17**
45			Test module integration	5 days	Tue 09-05-17	Sat 13-05-17
46			Identify anomalies to specifications	2 days	Sun 14-05-17	Mon 15-05-17
47			Modify code	3 days	Tue 16-05-17	Thu 18-05-17
48			Re-test modified code	2 days	Fri 19-05-17	Sat 20-05-17
49			Integration testing complete	0 days	Sat 20-05-17	Sat 20-05-17
50			**Documentation and Delivery**	**86.5 days**	**Thu 09-03-17**	**Sat 03-06-17**
51			**Documentation**	**48 days**	**Fri 17-03-17**	**Wed 03-05-17**



Project: Software Development
Date: Wed 15-03-17

Task	▬	Summary	▬	Inactive Milestone	◇	Duration-only	▬	Start-only	[	External Milestone	◆	Manual Progress	▬
Split	┈	Project Summary	▬	Inactive Summary	▬	Manual Summary Rollup	▬	Finish-only	]	Deadline	⬇		
Milestone	◆	Inactive Task	▭	Manual Task	▬	Manual Summary	▬	External Tasks	▬	Progress	▬		

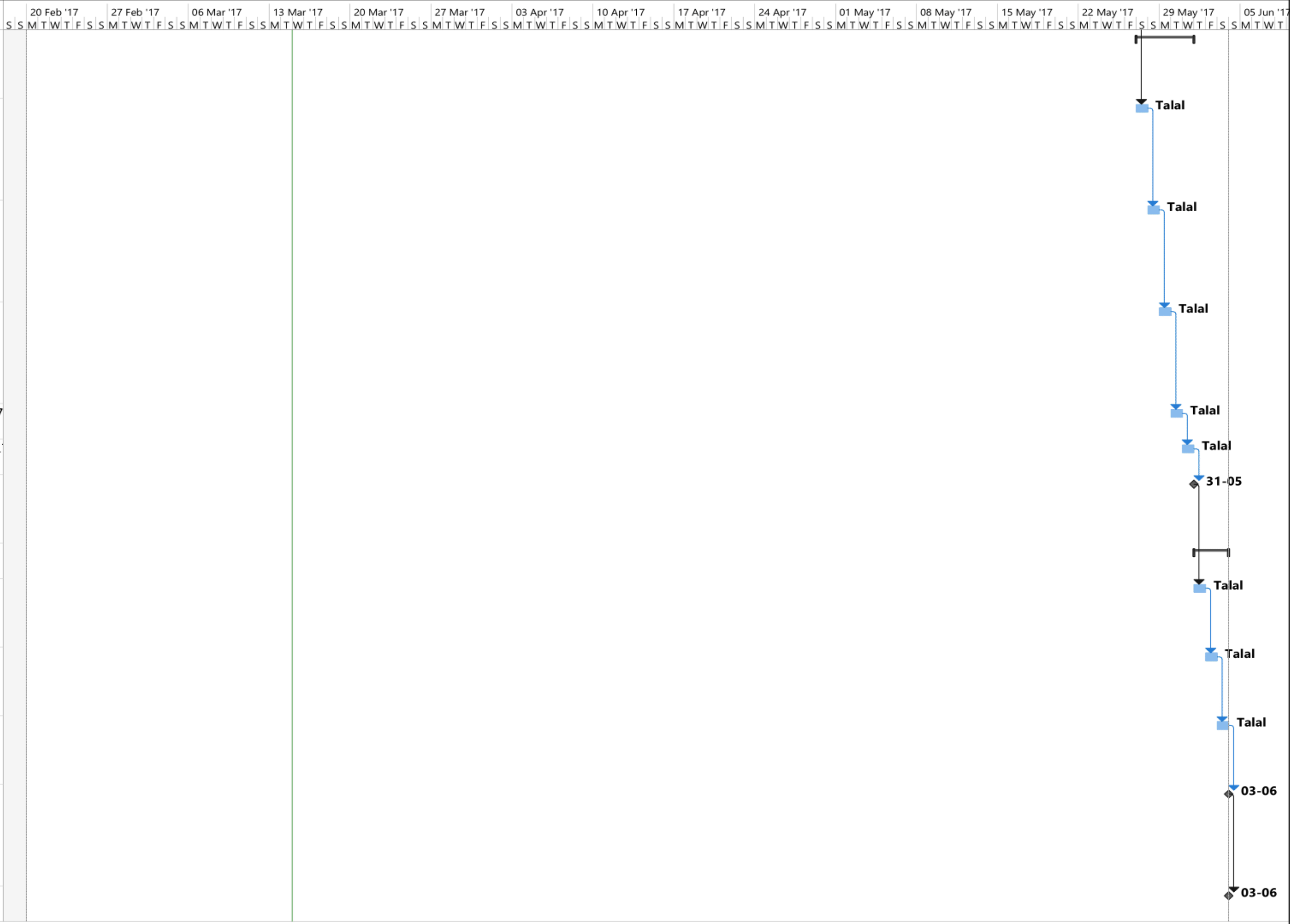
ID		Task Mode	Task Name	Duration	Start	Finish
52			Develop Help specification	1 day	Sat 01-04-17	Sat 01-04-17
53			Develop SRS Document	40 days	Fri 10-03-17	Tue 18-04-17
54			Develop SDS Document	16 days	Fri 17-03-17	Sat 01-04-17
55			Develop Help system	3 wks	Fri 14-04-17	Fri 28-04-17
56			Review Help documentation	3 days	Sat 29-04-17	Mon 01-05-17
57			Incorporate Help documentation feedback	2 days	Tue 02-05-17	Wed 03-05-17
58			Develop user manuals specifications	2 days	Sat 01-04-17	Sun 02-04-17
59			Develop user manuals	3 wks	Fri 14-04-17	Fri 28-04-17
60			Review all user documentation	2 days	Sat 29-04-17	Sun 30-04-17
61			Incorporate user documentation feedback	2 days	Mon 01-05-17	Tue 02-05-17
62			Documentation complete	0 days	Wed 03-05-17	Wed 03-05-17
63			**Pilot**	**78.5 days**	**Thu 09-03-17**	**Fri 26-05-17**
64			Identify test group	1 day	Thu 09-03-17	Fri 10-03-17
65			Develop software delivery mechanism	1.5 days	Fri 10-03-17	Sat 11-03-17
66			Install/deploy software to Google's Play Store	1 day	Sun 21-05-17	Sun 21-05-17
67			Obtain user feedback	4 days	Mon 22-05-17	Thu 25-05-17
68			Evaluate testing information	1 day	Fri 26-05-17	Fri 26-05-17
69			Pilot complete	0 days	Fri 26-05-17	Fri 26-05-17



Project: Software Development
Date: Wed 15-03-17

Task	▬	Summary	┗━┛	Inactive Milestone	◇	Duration-only	▬	Start-only	[	External Milestone	◆	Manual Progress	▬
Split	┈┈	Project Summary	┗━┛	Inactive Summary	┗━┛	Manual Summary Rollup	▬	Finish-only	]	Deadline	⬇		
Milestone	◆	Inactive Task	▭	Manual Task	▬	Manual Summary	┗━┛	External Tasks	▬	Progress	▬		

ID	ℹ	Task Mode	Task Name	Duration	Start	Finish
70			**Deployment**	**5 days**	**Sat 27-05-17**	**Wed 31-05-17**
71			Determine final deployment strategy	1 day	Sat 27-05-17	Sat 27-05-17
72			Develop deployment methodology	1 day	Sun 28-05-17	Sun 28-05-17
73			Secure deployment resources	1 day	Mon 29-05-17	Mon 29-05-17
74			Train support staff	1 day	Tue 30-05-17	Tue 30-05-17
75			Deploy software	1 day	Wed 31-05-1	Wed 31-05-1
76			Deployment complete	0 days	Wed 31-05-17	Wed 31-05-17
77			**Post Implementation Review**	**3 days**	**Thu 01-06-17**	**Sat 03-06-17**
78			Document lessons learned	1 day	Thu 01-06-17	Thu 01-06-17
79			Distribute to team members	1 day	Fri 02-06-17	Fri 02-06-17
80			Create software maintenance team	1 day	Sat 03-06-17	Sat 03-06-17
81			Post implementation review complete	0 days	Sat 03-06-17	Sat 03-06-17
82			Software development complete	0 days	Sat 03-06-17	Sat 03-06-17



Project: Software Development Date: Wed 15-03-17														
	Task	▬	Summary	▬	Inactive Milestone	◇	Duration-only	▬	Start-only	[	External Milestone	◆	Manual Progress	▬
	Split	┈	Project Summary	▬	Inactive Summary	▬	Manual Summary Rollup	▬	Finish-only	]	Deadline	⬇		
	Milestone	◆	Inactive Task	▭	Manual Task	▬	Manual Summary	▬	External Tasks	▬	Progress	▬		

2.8. List of Milestones

Table 7. List of Milestones

No.	Description of Output	Expected Time Interval	
1	Scope determination and approval	**20-02-17**	**23-02-17**
2	Analysis/Software Requirements	**23-02-17**	**09-03-17**
3	System Design (SDS Stage)	**09-03-17**	**31-03-17**
4	Software Development Stage	**01-04-17**	**23-04-17**
5	Unit Testing	**19-04-17**	**08-05-17**
6	Integration Testing	**08-05-17**	**20-05-17**
7	Documentation	**17-03-17**	**03-05-17**
8	Pilot	**09-03-17**	**26-05-17**
9	Deployment	**27-05-17**	**31-05-17**
10	Post Implementation Review	**01-06-17**	**03-06-17**

2.9. List of Risks

Table 8. List of Risks

Risk	Probability	Effects	Your Strategy
The time required to develop the software is underestimated.	High	Serious	The most important requirements of the project should always be implemented first. We will have more time later on to implement the non-important requirements.
Software tools cannot work together in an integrated way.	High	Tolerable	Always minimize the number of design tools used and make sure that the outputs of these tools are compatible with each other.
Customers fail to understand the impact of requirements changes.	Moderate	Tolerable	Conduct frequent meetings with the stakeholders and keep being updated on latest requirement changes.
The rate of defect repair is underestimated.	Moderate	Tolerable	Replace potentially defective components with more reliable bought-in components.
The size of the software is underestimated.	Moderate	Insignificant	Investigate buying software components; Investigate use of a program generator.
Code generated by code generation tools is inefficient.	Moderate	Insignificant	This risk is always expected since code generation tools often do not produce reliable code and this code always needs editing by the software developers.
Key staffs are ill at critical times in the project.	Moderate	Serious	Reorganize team so that there is more overlap of work and people therefore understand each other’s jobs.
The database used in the system cannot process as many transactions per second as expected.	Low	Serious	Investigate the possibility of buying a higher-performance database.

2.10. Commercialization Potential

Commercialization of our product can start as soon as the development and testing of the most important modules is done. However, to further guarantee that everything is working as planned and to have an advantage over other applications, we are going to wait until most of the features of the application are done. After that, the commercialization process starts when the application is uploaded to Google's Play Store. Then, many user feedbacks will be gained and improvements with new features and bug fixes will be implemented. In addition, during this time, an advertisement campaign will be made to promote our product and increase the user base. So the aim is to allow most of EMU Students to use this application.

2.11. Project Economic Expectations

Table 9. Project Economic Expectations

Time-to-market (month):	4
The expected increase in sales revenue (%):	25%
The expected increase in market share (%):	5%
Time to start to gain:	June 2017

2.12. Instrument / Equipment / Software / Release Purchases

Table 10. Instrument/Equipment/Software Purchases

Project Name	EMU Student Kit									
Line no	Instrument / Equipment / Software / Publication Name	No. of Item	Capacity	Technical specification	Purpose of Project Activities	Post-Project Place of Use / Purpose		Unit Price (USD)	Unit Price (TL)	Total Amount (TL)
						R & D	Production			
1	Android Studio	1		Android Integrated Development Environment (IDE)	Main IDE used for development of our project		Yes	-	-	-
2	Microsoft Project	1		Project Management Software	We will use this application to plan and schedule our project		Yes	589.99	2085.36	
3	Microsoft Office	1		An office suite of applications, servers, and services	Used in many areas of the project such as documentation		Yes	399.99	1413.80	

4	Microsoft Visio	1		Software Design Tool	Used to draw software design diagrams		Yes	299.99	1060.35	
5	Visual Paradigm	1		Software Design Tool and Code generator	Used to draw software design diagrams and generate code required for the application based on those diagrams		Yes	349	1233.54	
6	Mockflow Wireframe	1		User Interface Design Tool	Used to draw a User Interface for our system		Yes	208	735.38	
7	Gliffy	1		Software Design Tool	Used to draw software design diagrams		Yes	20	70.71	
8	GenMyModel	1		Software Design Tool	Used to draw software design diagrams		Yes	120	424.26	
									TOTAL	7023.4 TL

2.13. Quarterly Estimated Cost Form (TL)

Table 11. Quarterly Estimated Cost Form

Project Name :				
Cost Item	**2017**		**TOTAL (TL)**	**TOTAL COST RATE OF CONTENTS (%)**
	I	**II**		
Personnel	**20000**	**20000**	**40000**	**49.37**
Travel	**1000**	**1000**	**2000**	**2.47**
Instrument / Equipment / Software / Publications	**7023.4**		**7023.4**	**8.67**
Domestic Works Made By R & D and Testing Institutions	**1500**	**1500**	**3000**	**3.70**
International Works Made By R & D and Testing Institutions	**1600**	**1600**	**3200**	**3.95**
Domestic Services Procurement	**1800**	**1800**	**3600**	**4.44**
Overseas Service Procurement	**2000**	**2000**	**4000**	**4.94**
Material	**2100**	**2100**	**4200**	**5.18**
TOTAL COST	**37023.4**	**37000**	**81023.4**	**100**
CUMULATIVE COST				100
		IN THE PROJECT TOTAL MAN-MONTH	**720 hours**	

3. REQUIREMENTS ANALYSIS

3.1. Functional Requirements

3.1.1. User Requirements

R1.1: The user should be able to add, edit and delete his timetable.

R1.2: The user should be able to add, edit and delete his exams timetable.

R1.3: The user should be able to add, edit and delete courses.

R1.4: The user should be able to add, edit and delete text in the notes page.

R1.5: The user should be able to add and delete pictures in the notes page.

R1.6: The user should be able to add and delete voice recordings in the notes page.

R1.7: The user should be able to add or change the applications lock pin through the settings page.

R1.8: The user should be able to choose the desired notifications time frame through the settings page.

R1.9: The user should be able to choose the background of the application through the settings page.

R1.10: The user should be able to choose the color theme of the application through the settings page.

R1.11: The user should be able to choose what notifications he wants to receive through the settings page.

R1.12: The user should be able to input current grades and calculate GPA/CGPA in GPA calculator page.

R1.13: The user should be able to send direct messages, and complaints to the admin through the contact us page.

R1.14: The user should be able to navigate to the desired page through the main menu.

R1.15: The user should be able to view the user manual in help page.

3.1.2. Admin Requirements

R2.1: The admin should be able to add, edit and delete events in the academic calendar.

R2.2: The admin should be able to edit the interactive map’s locations and places of interest.

R2.3: The admin should be able to add, edit and delete the bus schedules.

R2.4: The admin should be able to add, edit and delete phone numbers in the information page.

R2.5: The admin should be able to add, edit and delete office hours in information page.

R2.6: The admin should be able to add, edit and delete RSS feed links in the news page.

R2.7: The admin should be able to add, edit and delete contents in the help page.

R2.8: The admin should be able to add, edit and delete contents in the about page.

R2.9: The admin should be able to add, edit and delete contents in the contact us page.

R2.10: The admin should be able to add, edit and delete settings options.

3.1.3. System Requirements

R3.1: The user should be able to see and interact with a login page with a pin lock prompt when opening the application.

R3.2: The home page should contain buttons linked to the following pages:

Timetables	Calendar	Map
Bus Schedule	News	Notes
Info	GPA Calculator	Tasks
Settings	Quick add	

R3.3: The timetables page should contain a drop down menu to select “My timetable”, “Instructor timetable” and “Exams timetable”.

R3.4: The calendar page should contain a drop down menu to select different academic calendar.

R3.5: The map page should contain a similar interactive map to that on EMU’s website.

R3.6: The bus schedule page should contain bus schedules as well as bus route maps.

R3.7: The news page should contain RSS feed up to date with news tab in EMU’s website.

R3.8: The notes page should contain an applicable interface to input notes, pictures and voice notes for the selected course.

R3.9: The info page should contain phone numbers, opening hours and announcements for points of interest in the university campus.

R3.10: The info page should contain emergency phone numbers in TRNC.

R3.11: The GPA calculator page should contain an offline GPA and CGPA calculator for students to use.

R3.12: The tasks page should contain task the student has assigned for courses.

R3.13: The tasks page should contain a drop down menu to group tasks by course.

R3.14: Each task in tasks page should contain a checkbox to mark when done.

R3.15: The settings page should contain button links to the pages: about, help and contact us.

R3.16: The settings page should contain the option to change the login page.

R3.17: The settings page should contain the option to change the theme color of the application.

R3.18: The settings page should contain the options to change the background of the application.

R3.19: The about page should contain the description of the application as well as the version of the application.

R3.20: The help page should contain easy and user friendly steps to use the application.

R3.21: The contact us page should contain a mailto:// link to the admin.

R3.22: The quick add button should open a menu to quickly add tasks, notes, pictures and courses.

R3.23: Each page should contain a header banner that includes the page title.

R3.24: The system should show an error message if the user enters a wrong PIN.

R3.25: The system should retry PIN after each wrong input.

R3.26: The system should proceed to the home page after PIN is matched with predefined PIN.

R3.27: The system should navigate to “Time Tables” page when clicked on.

R3.28: The system should navigate to “Calendar” page when clicked on.

R3.29: The system should navigate to “Map” page when clicked on.

R3.30: The system should navigate to “Bus Schedule” page when clicked on.

R3.31: The system should navigate to “News” page when clicked on.

R3.32: The system should navigate to “Notes” page when clicked on.

R3.33: The system should navigate to “Info” page when clicked on.

R3.34: The system should navigate to “GPA Calculator” page when clicked on.

R3.35: The system should navigate to “Settings” page when clicked on.

R3.36: The “Time Tables” page should contain a drop down menu at the top to choose from courses and exams.

R3.37: The Courses Time Table should show week days as buttons.

R3.38: Each week day button should navigate the user to the rotational day calendar.

R3.39: Each course in the Courses Time Table page should contain the course title, start time and end time.

R3.40: Clicking on a course shows a new page in which the user can edit the course title, date and time. As well as add notes.

R3.41: Exams Time Table page should show pre-added exams and their dates, times, course title associated and location of exam.

R3.42: The quick add button should be present in all pages of the system except the login page.

3.2. Non-Functional Requirements

3.2.1. Availability Requirements

The EMU Students Kit has been specifically designed as an Android Application and can only be run on Android embedded OS with API version 19 (aka) Android 4.4 Kit-Kat and above due to compatibility issues. Also, Mobile devices should have a working WIFI adapters and/or 3G network connection since the application acquires database and other updates using internet connection.

3.2.2. Performance Requirements

The application shall use the performance requirements of Android Application (APK) standards for Android API Version 19 and above.

3.2.3. Safety Requirements

The application should not contain any safety threats, malware or adware to users.

3.2.4. Implementation Requirements

The system should be implemented using the Java Programming language for Android 4.4 OS (Android Kitkat).

3.2.5. Security Requirements

The system shall be a very secure system and it should implement the Advanced Encryption Standard (AES) as set out by the U.S. National Institute of Standards and Technology (NIST) in 2001 to secure the passwords and the information within the database of the application. Only the people authorized to access the system shall be able to access it.

- AES encryption.
- HTTPS Database encryption
- Application Login PIN.
- Secure & encrypted APK file

3.3 Ethical issues

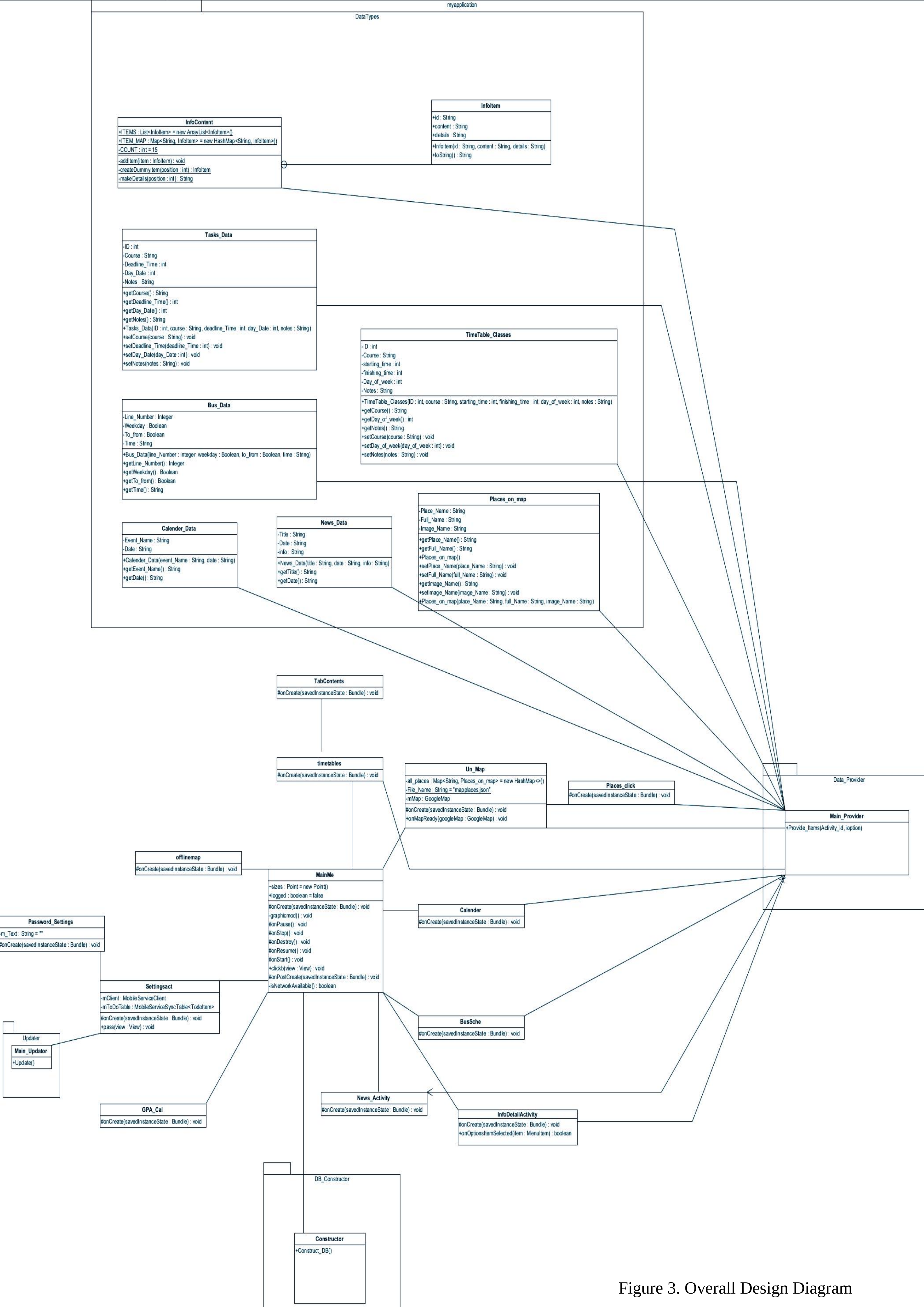
The user of this system may store some sensitive information such as his/her student portal password, bank account details, etc. Another unauthorized user may access this sensitive information. That’s why the application has a security feature where the user may choose to protect the application with an AES encrypted password. A student may also use this application to gain an unfair advantage/cheat during an exam.

4. DESIGN

4.1. High level design (architectural)

4.1.1. 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 4.2.2. An overall design diagram will be shown in Figure 3. This diagram was created using the Visual Paradigm Tool.



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
News_Activity
#onCreate(savedInstanceState : Bundle) : void
InfoDetailActivity
#onCreate(savedInstanceState : Bundle) : void
+onOptionsItemSelected(item : MenuItem) : boolean
GPA_Cal
#onCreate(savedInstanceState : Bundle) : void
DB_Constructor
Constructor
+Construct_DB()

Figure 3. Overall Design Diagram

4.1.1.1. Module Decomposition

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

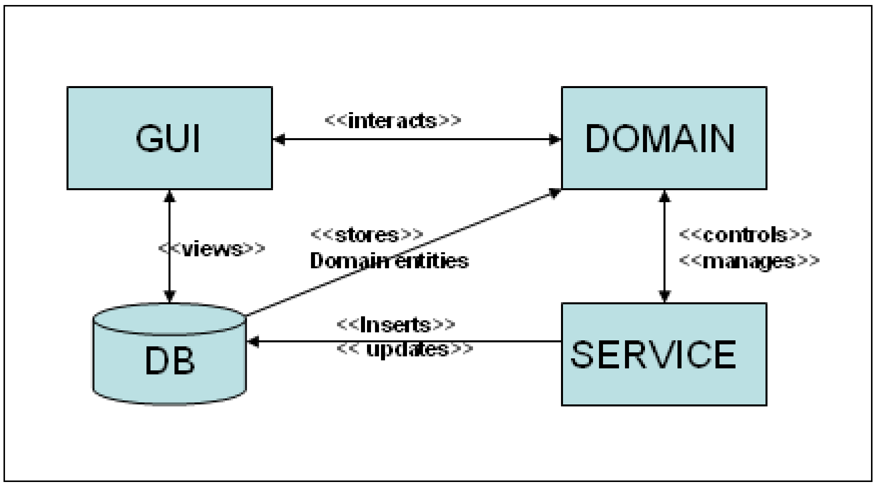


Fig 4. Components of EMU Student Kit

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 4.2.4.

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 4.2.2.

Module Entities

1. MainMe: The entity that manages the main menu and Android button clicks activities.
2. timetables: The entity that manages the Time Tables and Tasks module.
3. Calendar: The entity that manages the university calendar module.
4. BusSche: The entity that manages the bus schedules module.
5. News_Activity: The entity that manages the university news module.
6. GPA_cal: The entity responsible for calculating the GPA/CGPA.
7. Settingsact: The entity responsible for managing many of the system's settings.
8. Constructor: The entity responsible for creating a database when it is not available.
9. Un_Map: The entity responsible for managing the interactive map.
10. Password_Settings: The entity responsible for managing the PIN code.

4.1.1.2. Concurrent Process Decomposition

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

4.1.1.3. 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 5 to 7. These Figures were created using Microsoft Visio Tool.

1. Places_on_map: entity holding all the details of the places on the interactive map.
2. News_Data: entity containing all the university news details.
3. Calendar_Data: entity containing all the university calendar details.
4. Bus_Data: entity containing all the bus related data.
5. TimeTable_Classes: entity containing the time table related data.
6. Tasks_Data: entity that contains the tasks related data.
7. InfoContent: entity that creates some sample university related information.
8. Infoitems: entity that contains the university related information.

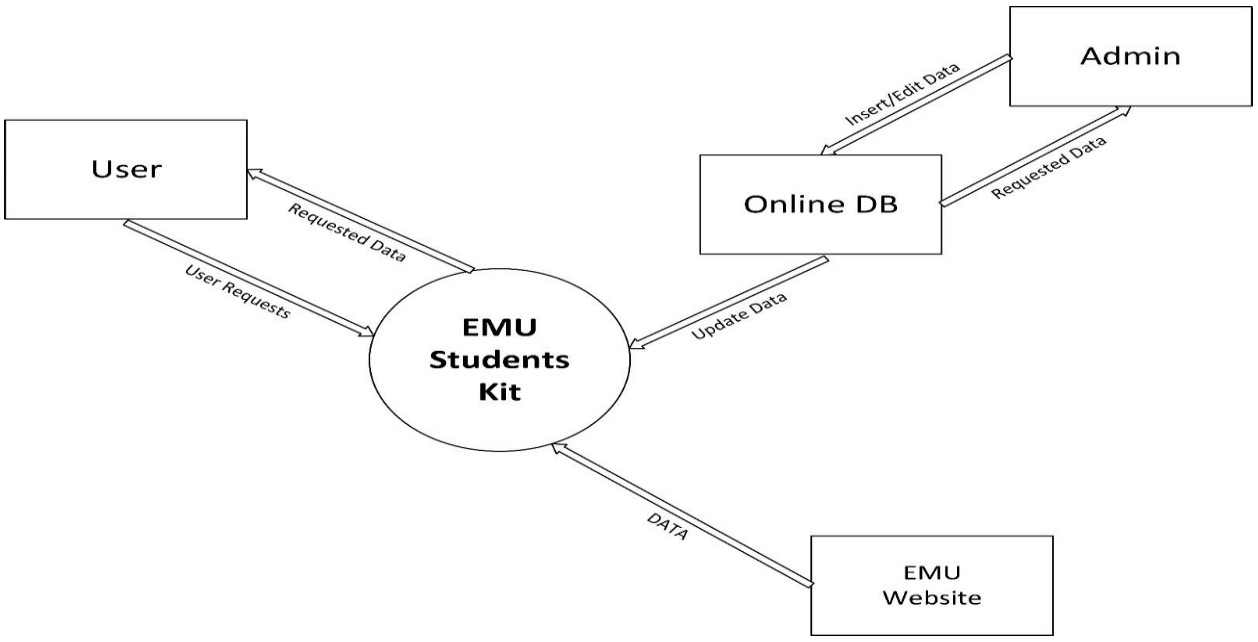


Fig 5. Context Diagram

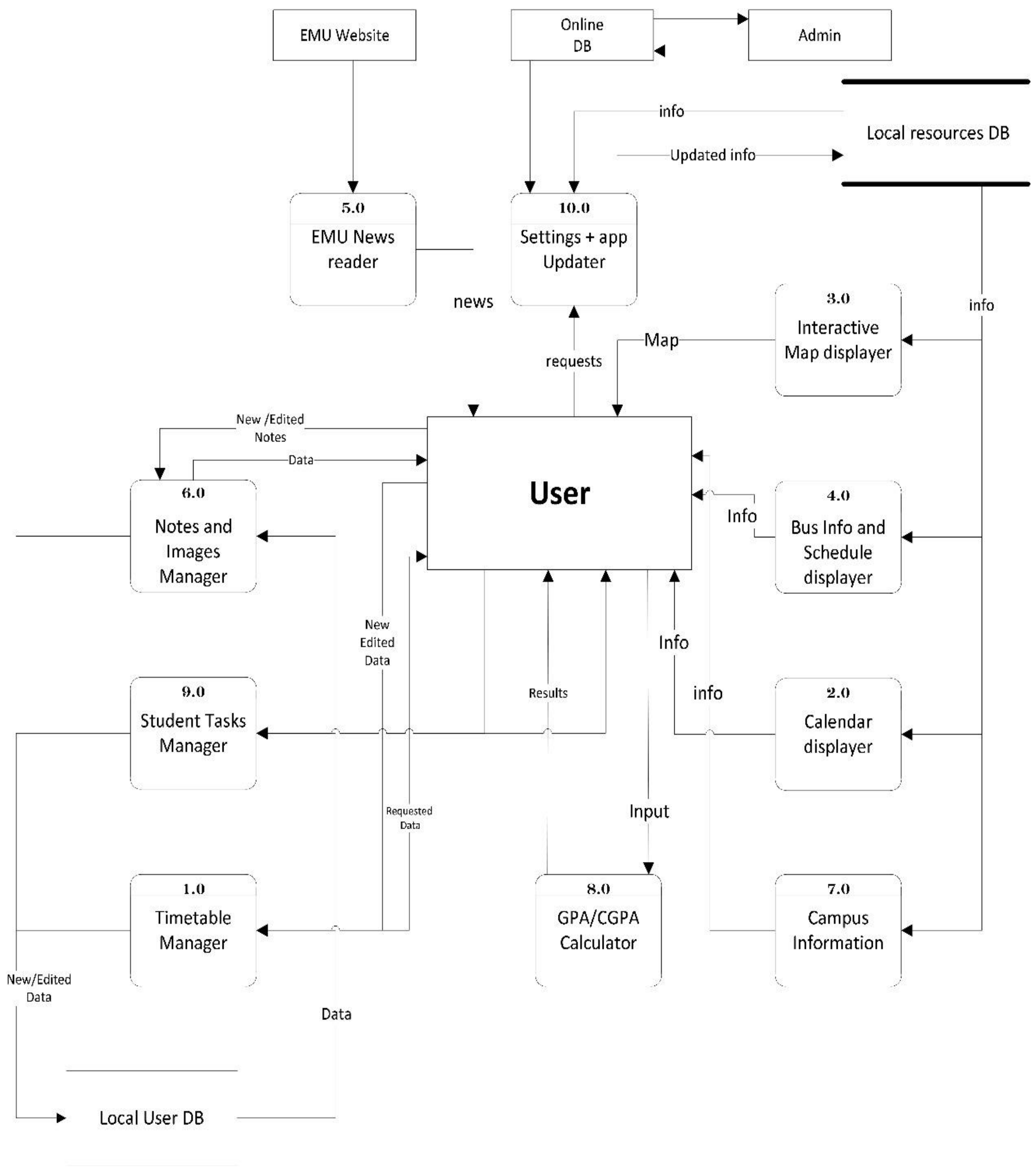


Fig 6. Level 0 Data Flow Diagram

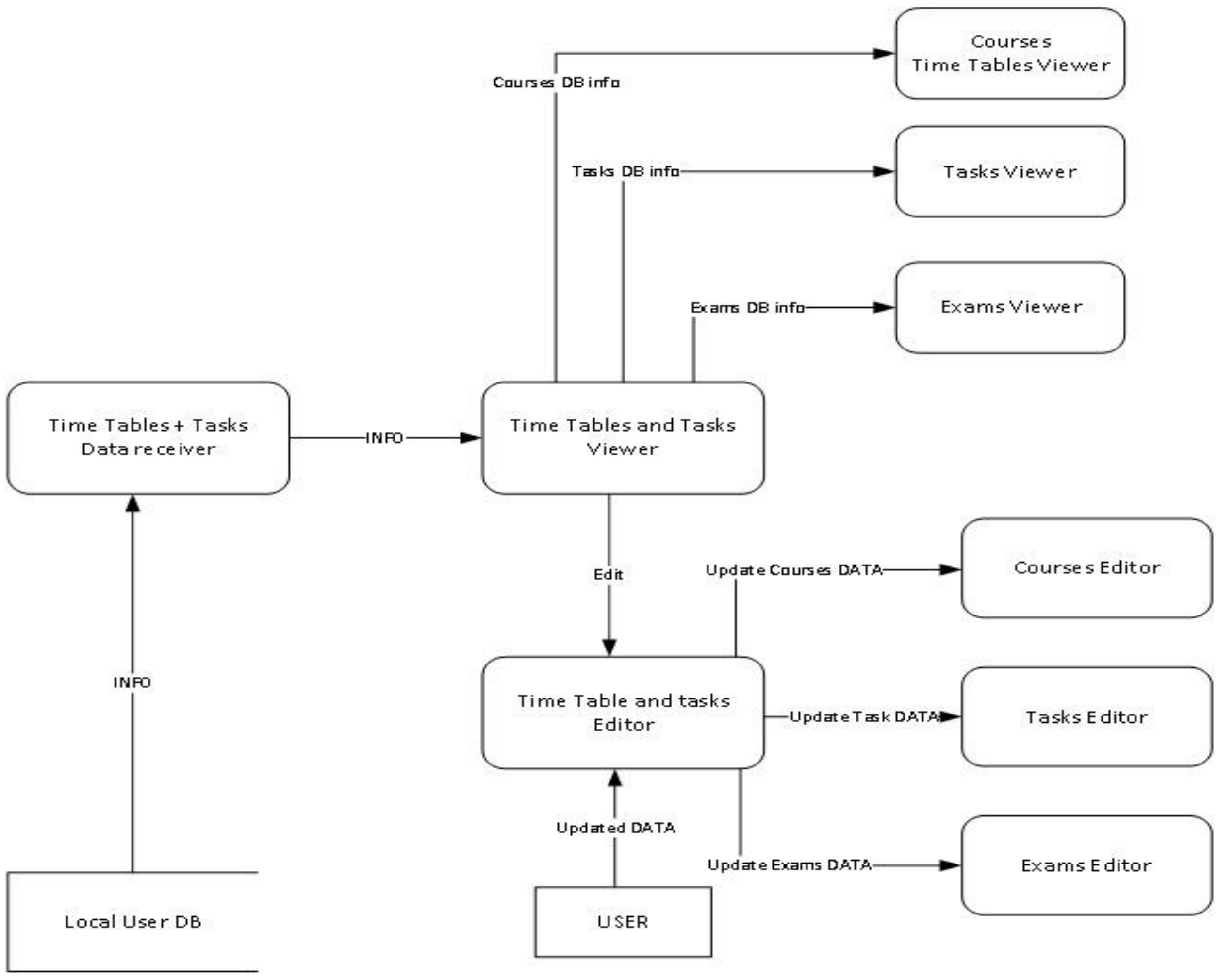


Fig 7. Level 1 Data Flow Diagram (Time Tables)

4.1.2. User Interface Design

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.

4.1.2.1. 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 8.

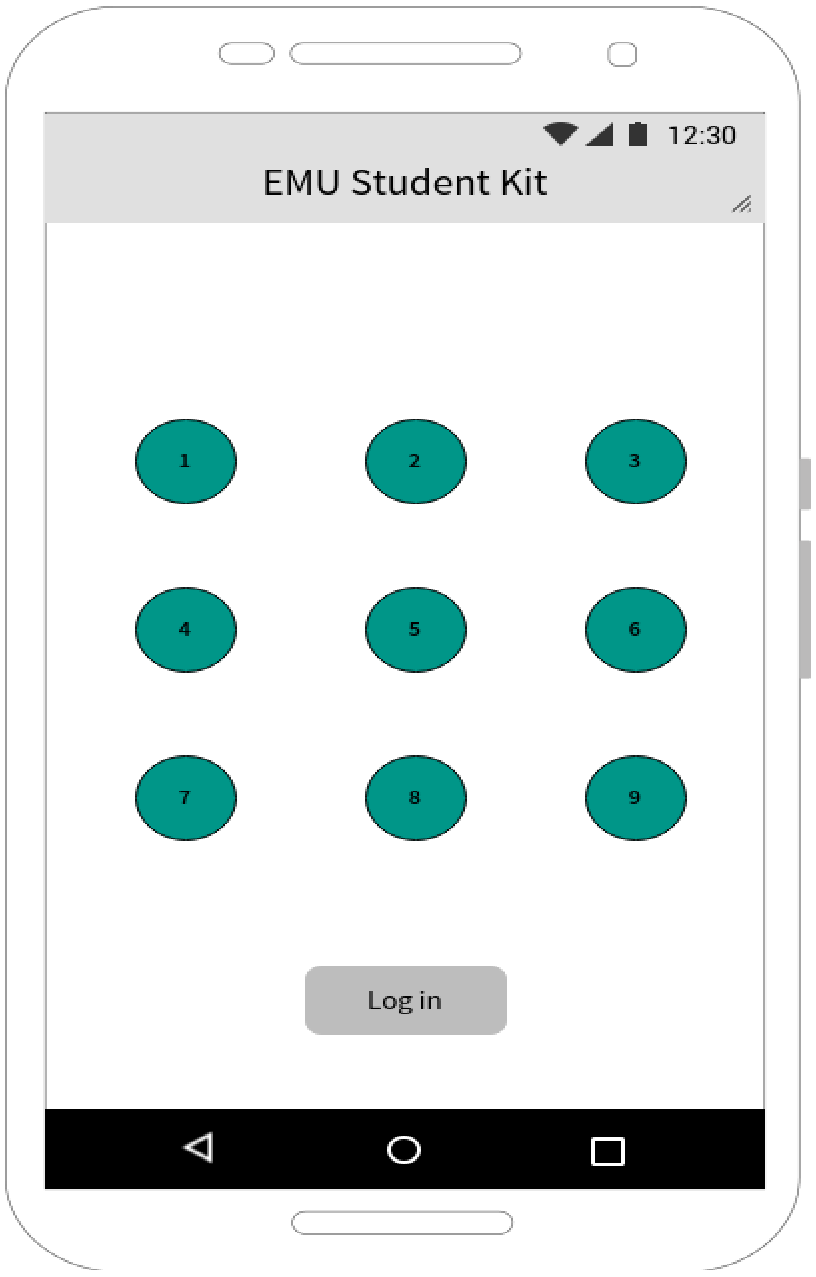


Fig 8. Login Screen

4.1.2.2. Main Menu

This is the main interface where the user can choose what he wishes to do. These functions are shown in Figure 9. Also, the user can click on the "+" sign to perform quick tasks such adding a new note or a new course.

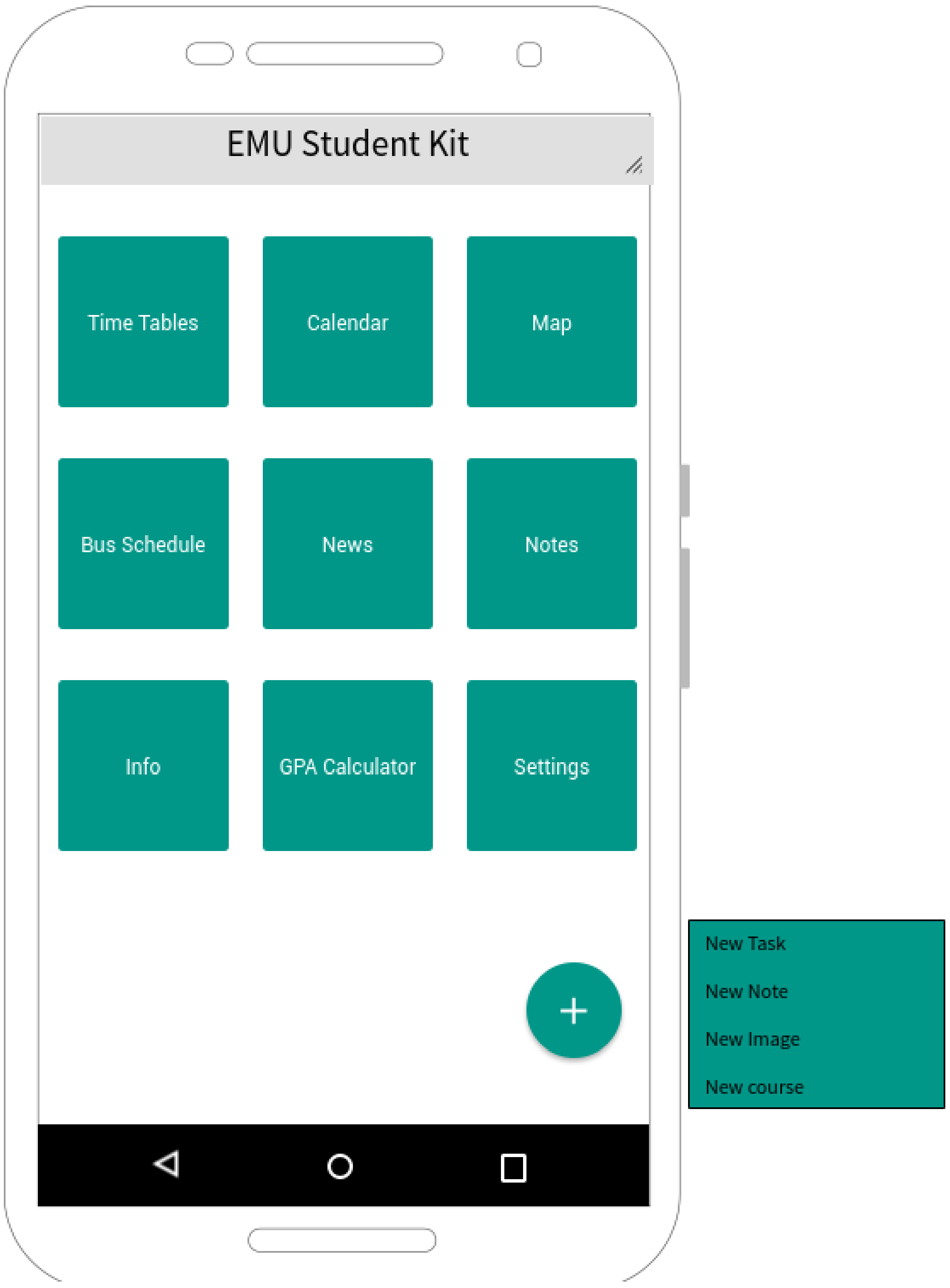


Fig 9. Main Menu

4.1.2.3 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 10 to 13.

4.1.2.3.1. Time Tables – Courses

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

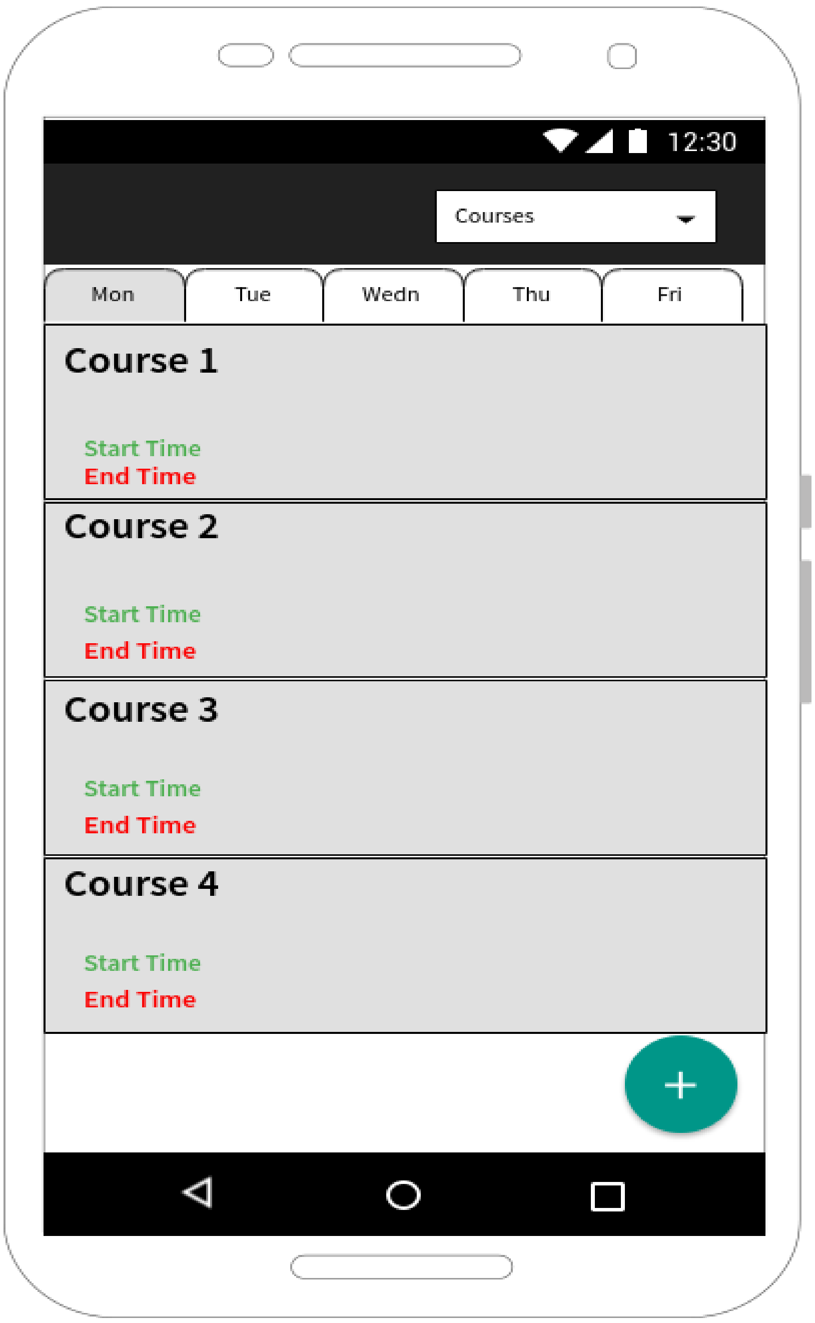


Fig 10. Courses section of the Time Tables

4.1.2.3.2. 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.

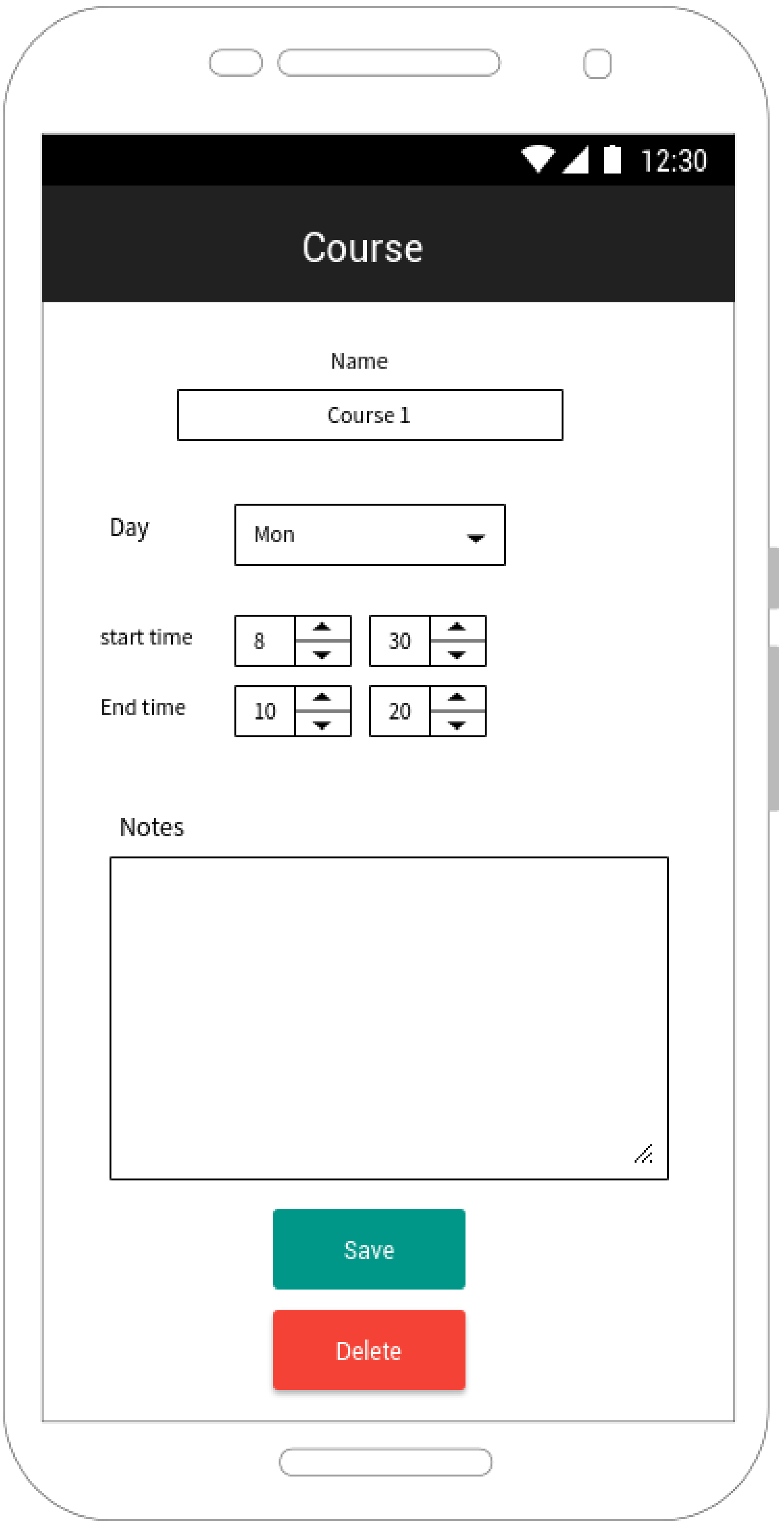


Fig 11. Editing Course 1 from the list of courses created by the user

4.1.2.3.3. Time Tables – Exams

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

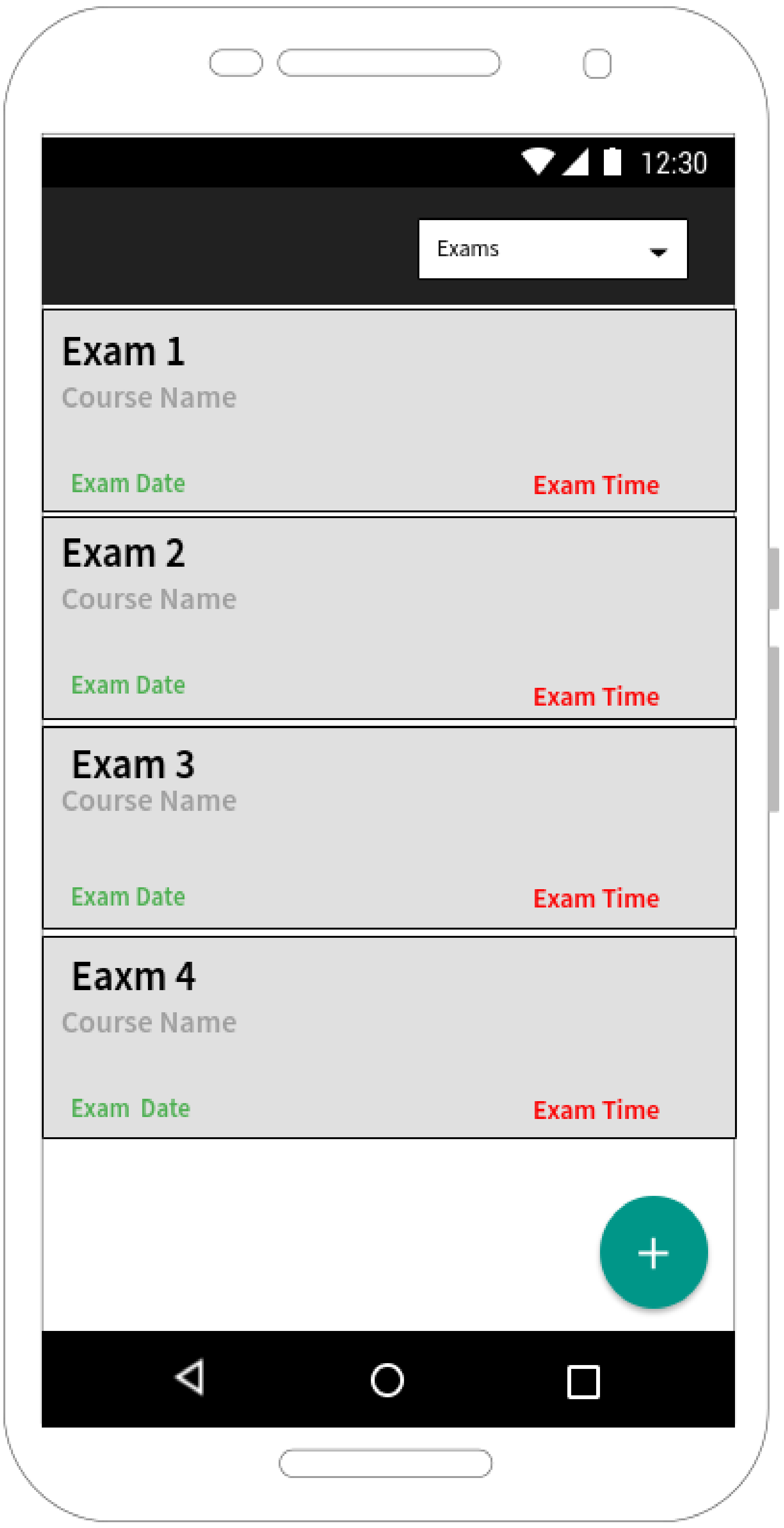


Fig12. Exams section of the Time Tables

4.1.2.3.4. Time Tables – Tasks

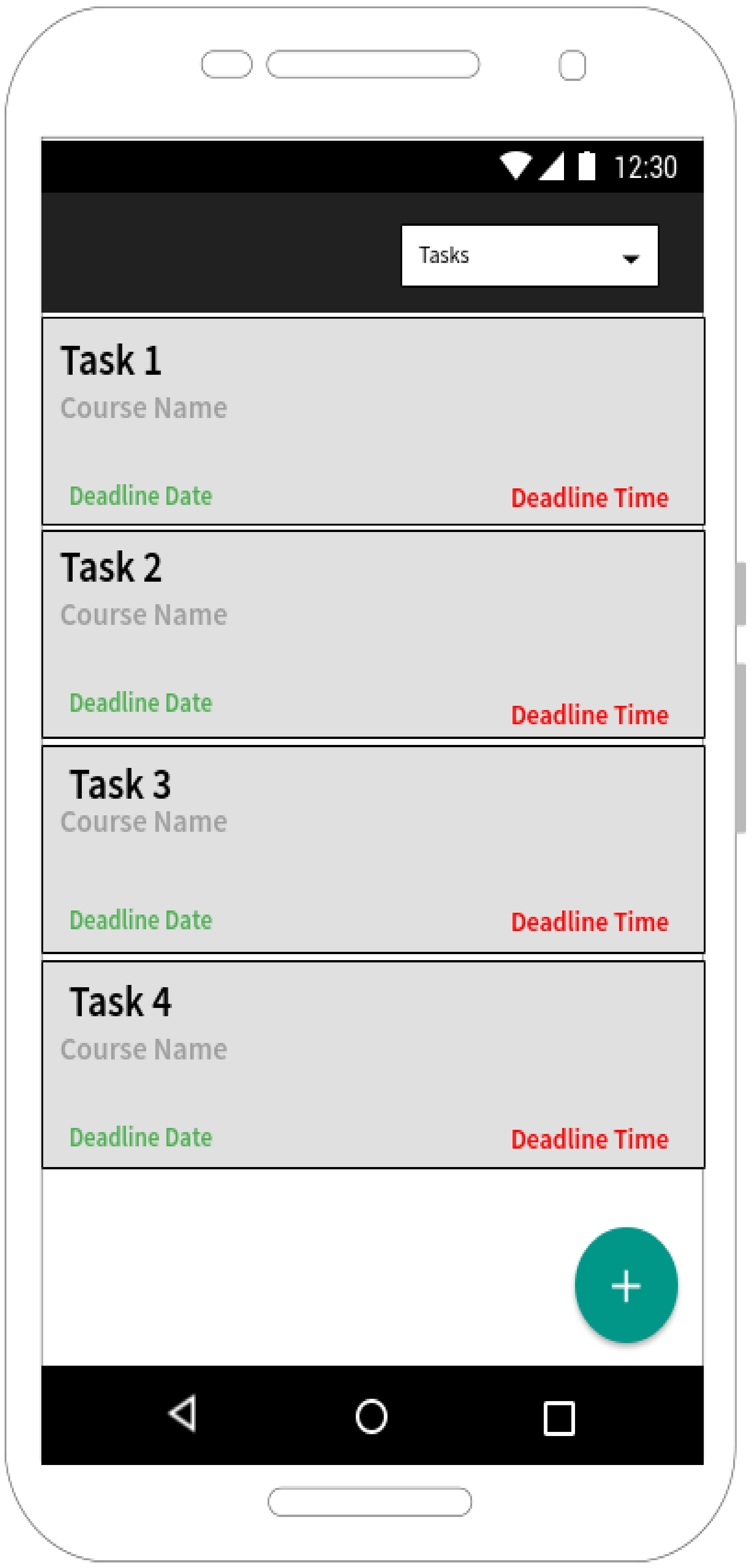


Fig 13. Tasks section of the Time Tables

4.1.2.4. Bus Schedules

This shows all routes, timings and operating days of the buses of EMU. This is shown in Figure 14.

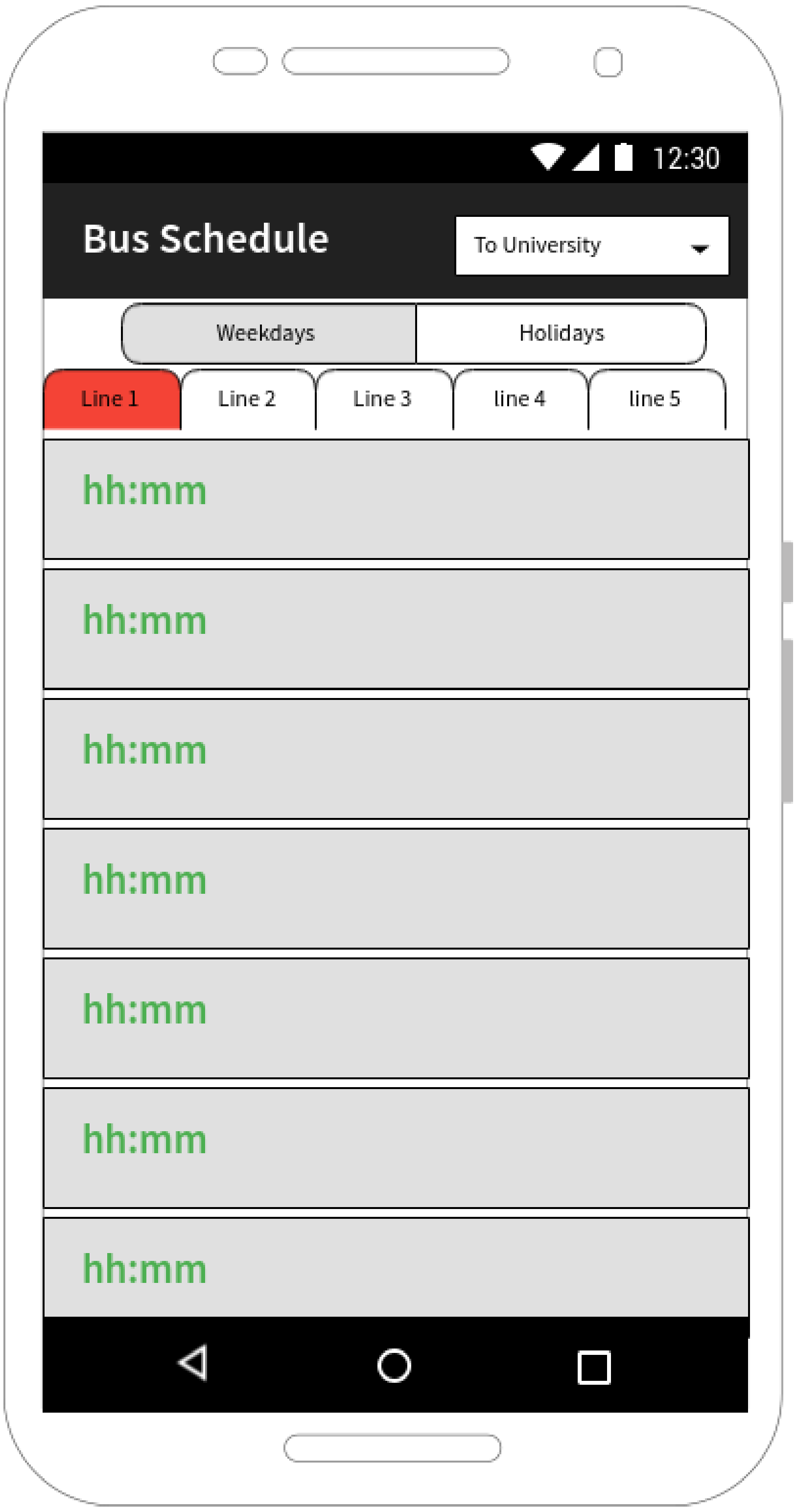


Fig 14. Bus Schedules Interface

4.1.2.5. News

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

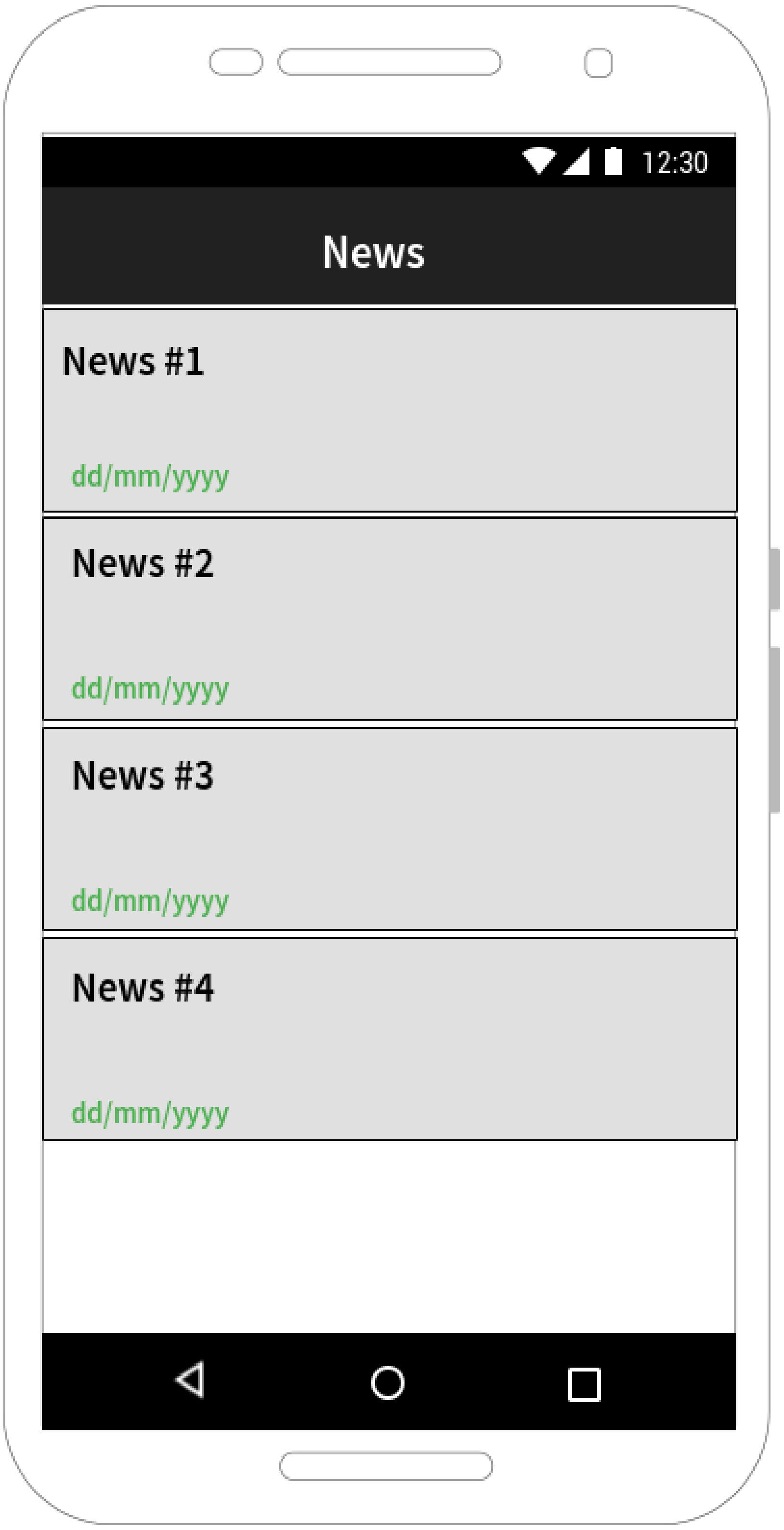


Fig 15. News Interface

4.1.2.6. 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 16.

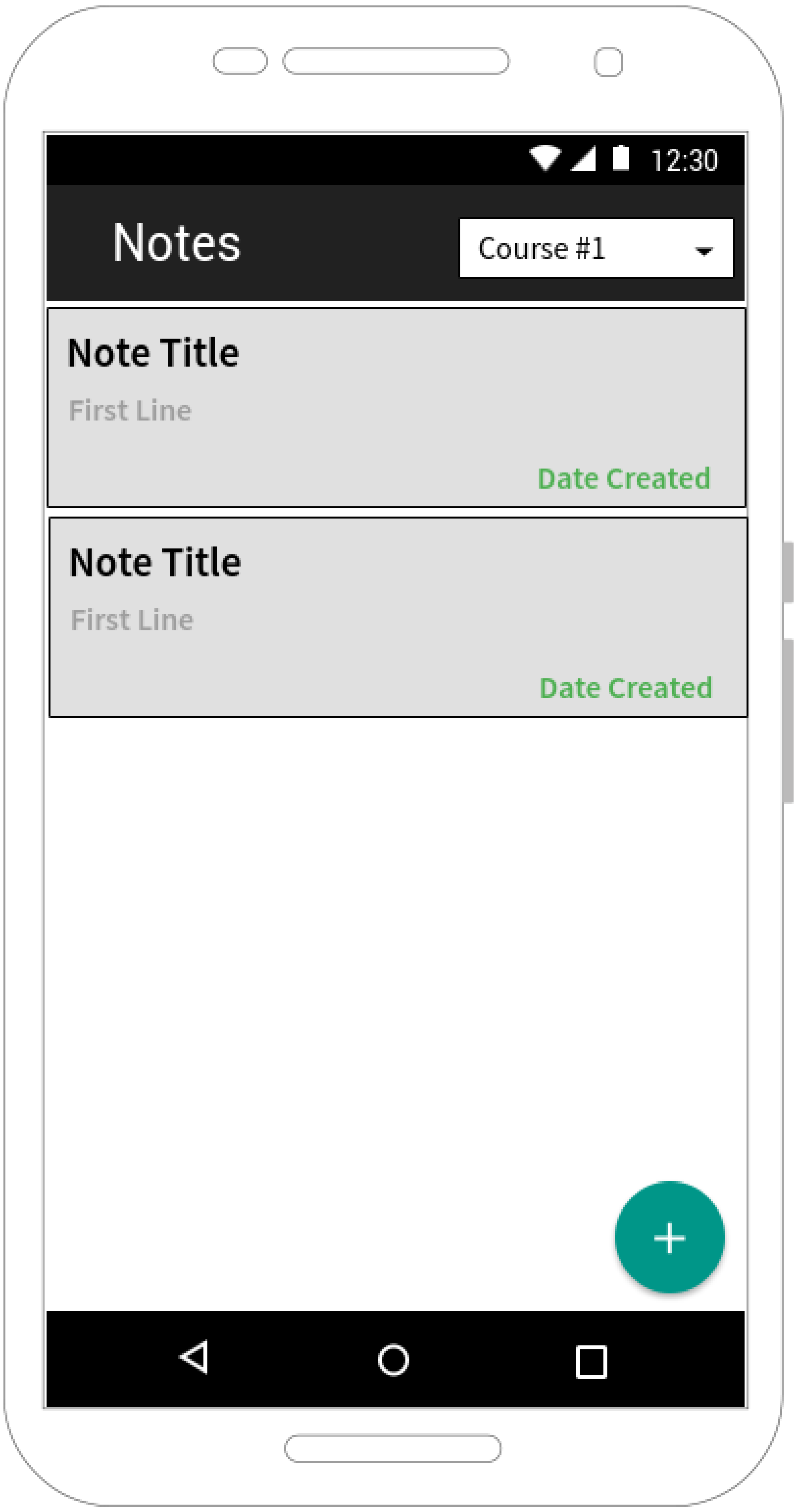


Fig 16. Notes Interface

4.1.2.7. 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.

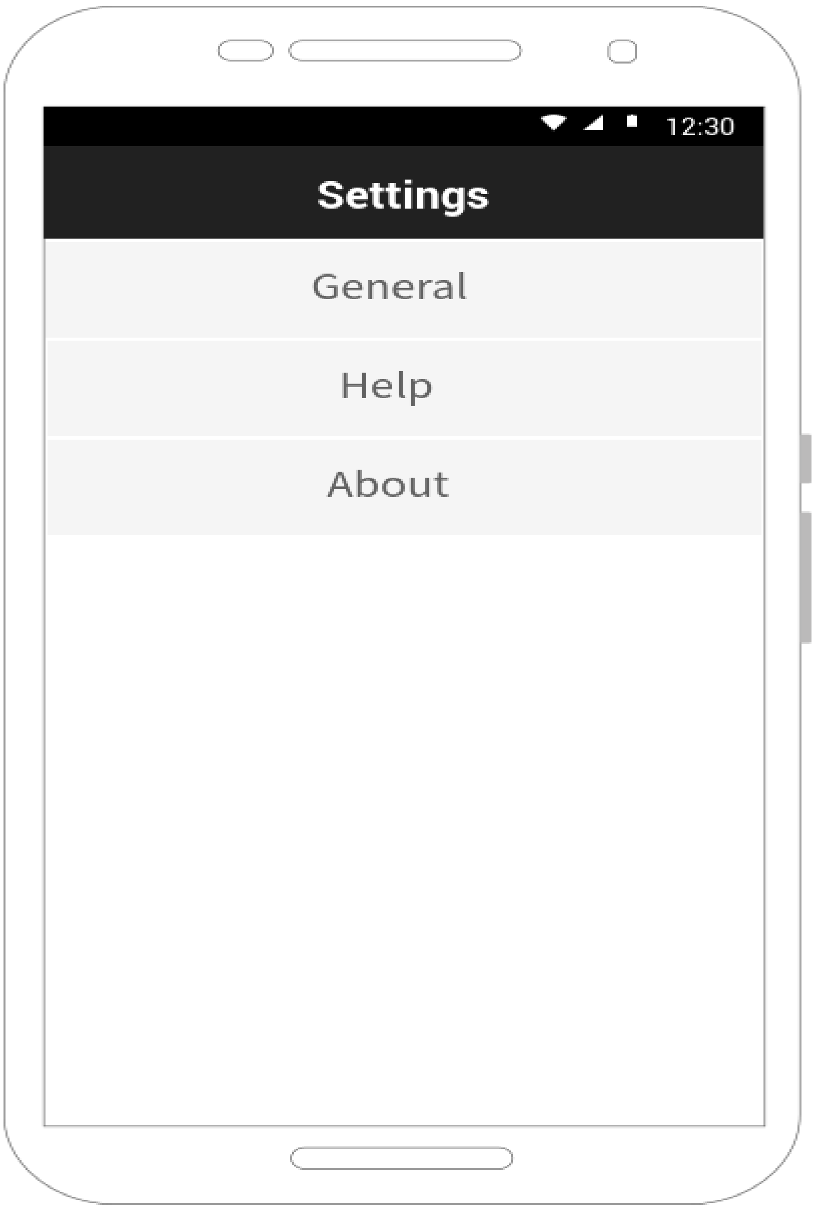


Fig 17. Settings page

4.1.2.8. Calendar

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

4.1.2.9. 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.

4.1.2.10. 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.

4.1.2.11. GPA Calculator

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

4.2. Low level design (components used)

4.2.1. Dependency Description

4.2.1.1. Intermodule Dependencies

As indicated in Figure 4, 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.

4.2.1.2. Data Dependencies

This will be represented as an Entity Relationship Diagram in Figure 18. This diagram was created using the Gliffy tool.

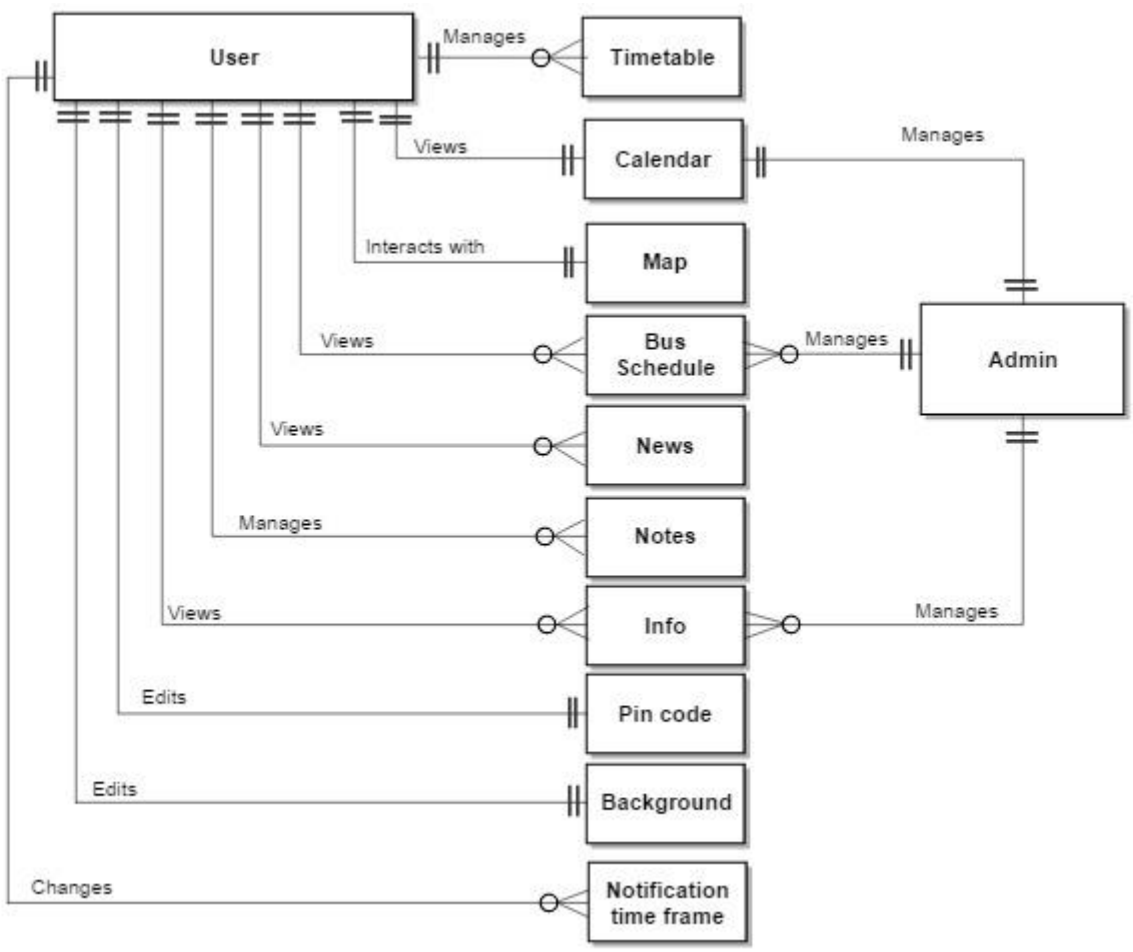


Fig 18. ERD for EMU Student Kit

4.2.2. Detailed Design

4.2.2.1. 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 Tool will be shown in Figure 19.

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

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

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.

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

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

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.

Main_Updater:

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

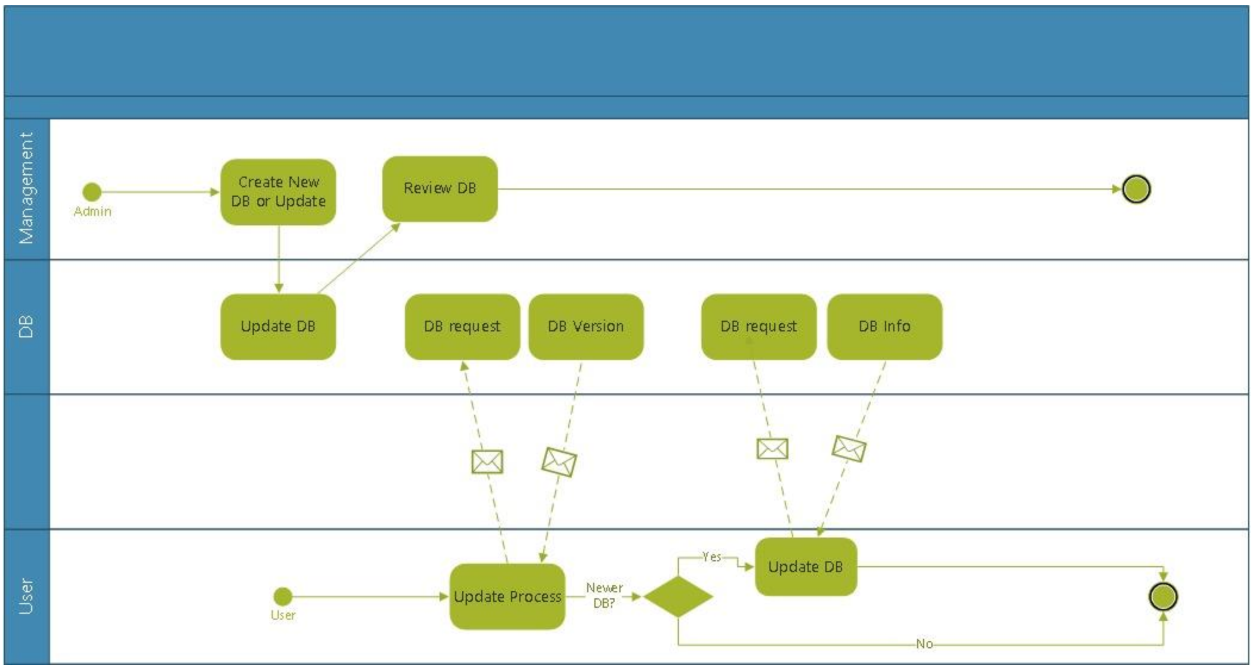


Fig 19. Business Process Model

4.2.2.2. Data Detailed Design

Some important data modules are described in this Section.

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

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

4.3. UML Interaction Diagrams

A use case diagram which was created using the GenMyModel tool is shown in figure 20. The two sequence diagrams shown in Figures 21 and 22 were created using the Gliffy tool. Also, two activity diagrams are shown in Figures 23 and 24. These were created using the GenMyModel tool by referring to some ideas from creately.com[17] [18] and slideshare.com[19] websites.

4.3.1. Use Case Diagram

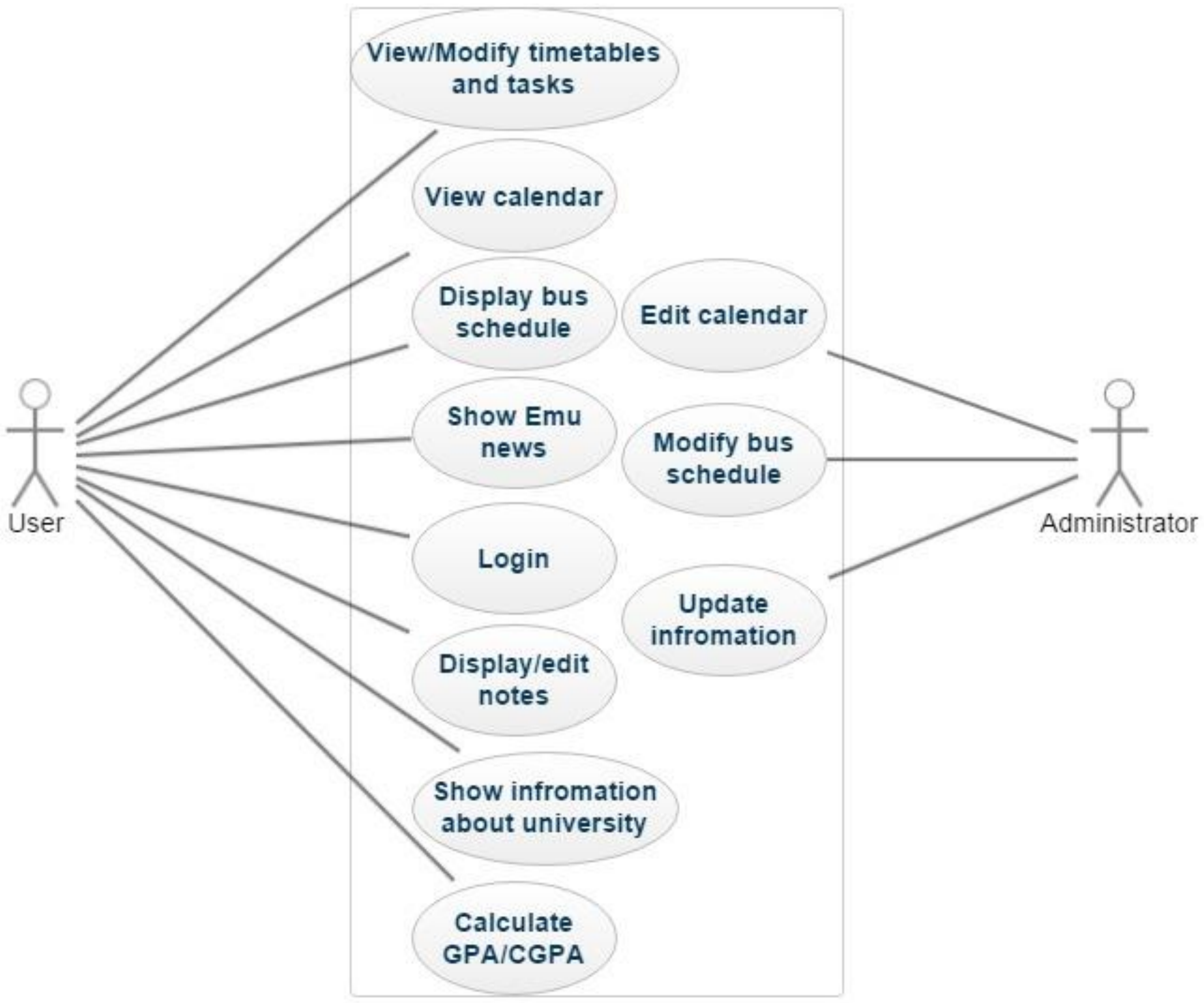


Fig 20. Use Case Diagram

4.3.2. Sequence Diagram - Login

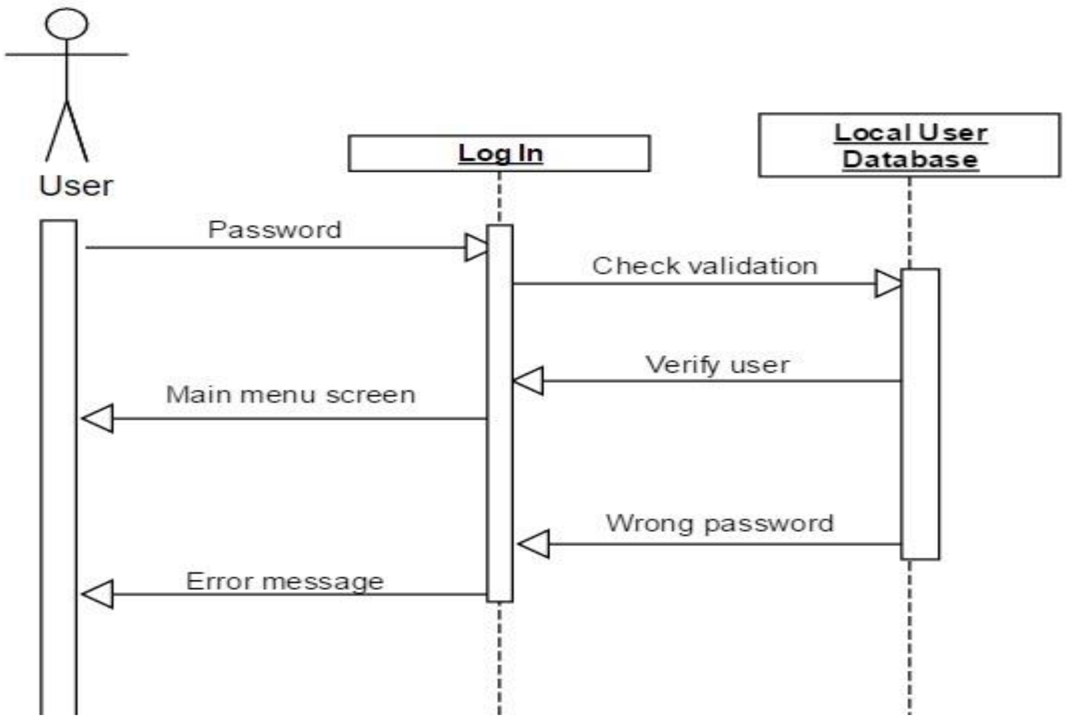


Fig 21. UML Sequence Diagram for User Login to System

4.3.3. Sequence Diagram – Add New Course

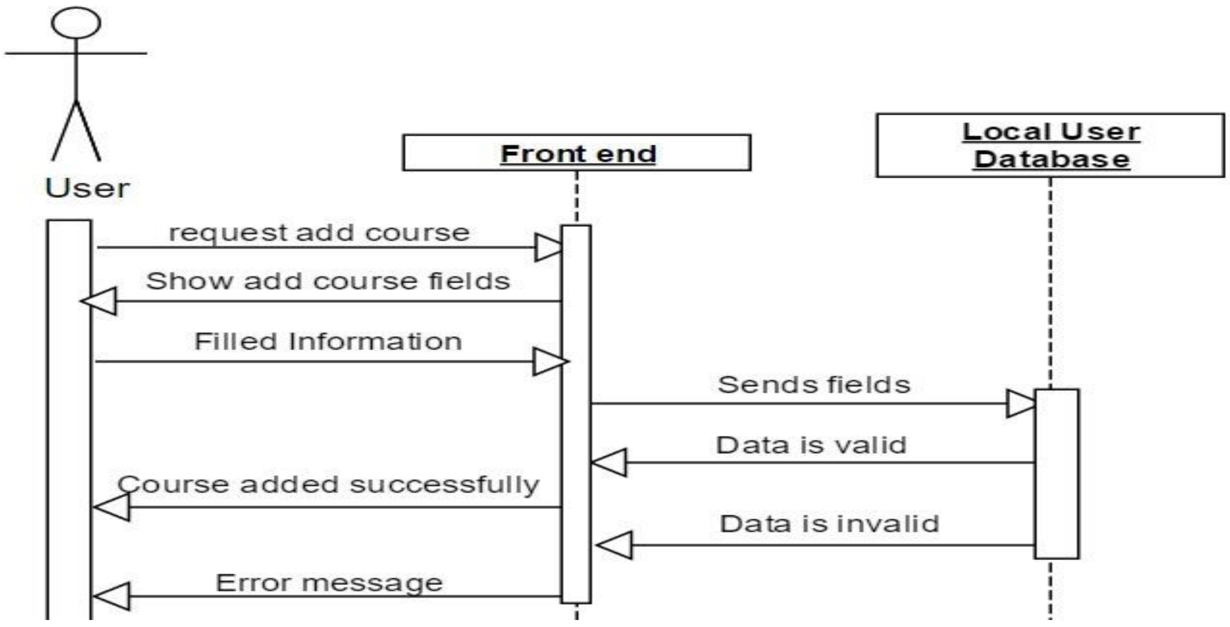


Fig 22. UML Sequence Diagram to Add a new Course

4.3.4. Activity Diagram – Login

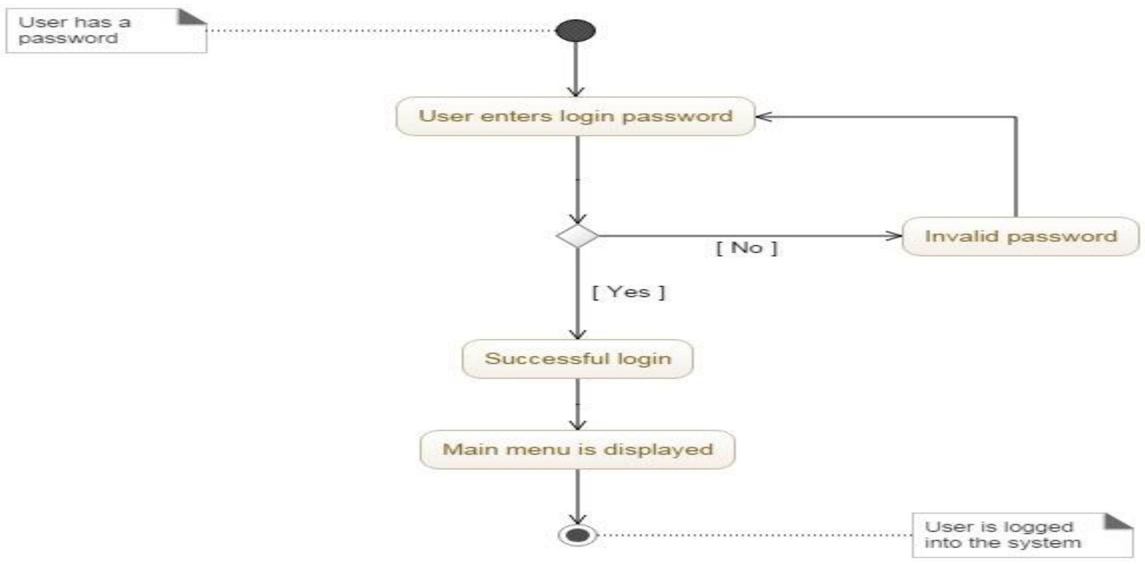


Fig 23. UML Activity Diagram for User Login to System

4.3.5. Activity Diagram – Manage Tasks

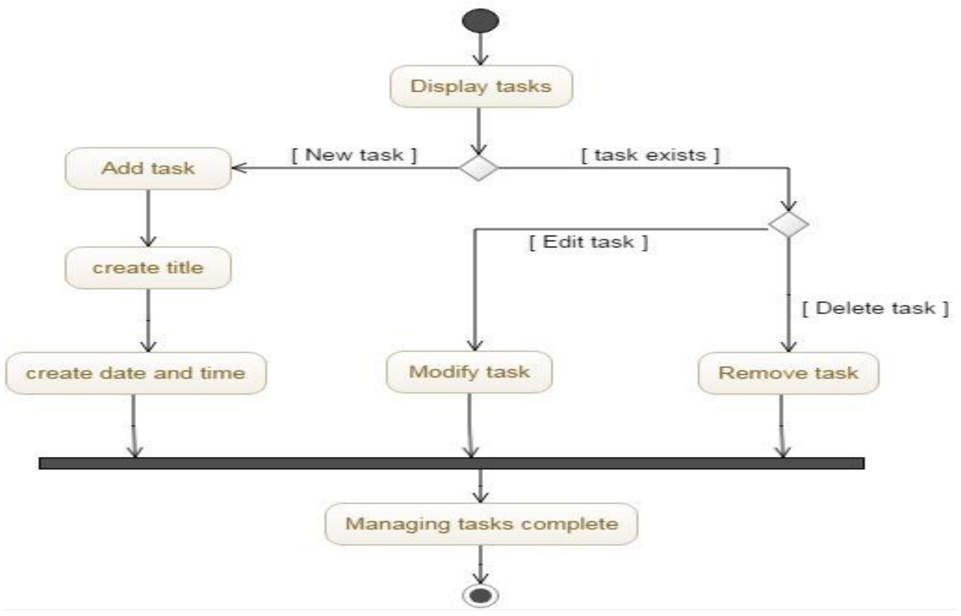


Fig 24. UML Activity Diagram for Managing Tasks

5. IMPLEMENTATION

5.1. Tools, technologies and platforms used

Java Programming language, Microsoft SQL Server Management Studio 2014, Android Studio IDE, Microsoft Azure, SQLite, JSON, Notepad++, DB Browser for SQLite

5.2. Use of Software Engineering Process Steps

These steps were carried out in order to build the project:

1. Determination of an ideal Software Development Approach: Incremental Model. This was discussed in Section 2.6 and is shown in figure 25.

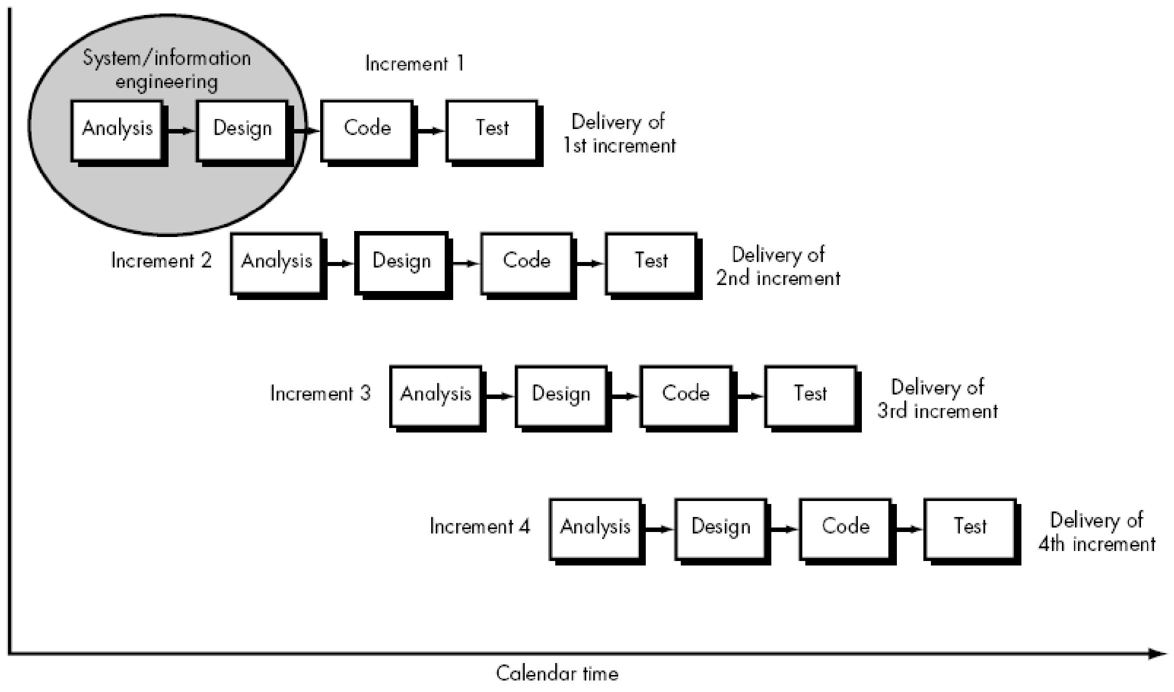


Figure 25. Incremental Method of Development

2. Make a Software Development Plan with the help of Microsoft Project tool.

3. Arrange the features of the system according to their importance.
   High Priority:
   a. Time Tables
   b. Bus Schedules
   c. Interactive Map
   d. Password

Low Priority:
e. Notes
f. Info
g. Calendar
h. News
i. GPA/CGPA Calculator
j. Personalization Settings

4. Starting with the most important modules, write their specific requirements and prepare the SRS.

5. At the same time, prepare the architecture of the system and overall design and write the SDS. Also, prepare the designs for the important modules to be handed over to the programmers using the Requirements made by the Requirements Engineers.

6. When the requirements and design of the important modules are over, start the development of the most important modules.

7. Begin the development of the next modules by the time the testing of the already developed modules is over.

8. The steps above are repeated until development of all modules is complete and the system is integrated.

9. The system as a whole is verified and validated and then deployed.

10. Updates and Maintenance to the system is periodically done.

5.3. Algorithms

```
Algorithm 1: Login
Begin
Enter password
if password correct then log user in
else output "wrong password"
End
```

```
Algorithm 2: Update data
Begin
```

```
if user chooses Settings -> UPDATE then update app data
else use old database data
End
```

Algorithm 3: Add lecture/Task/Exam

```
Begin
user chooses TimeTables -> Lectures/Tasks/Exams -> Add New -> fills data
if user clicks save then save lecture/task/exam in database
else do nothing
End
```

Algorithm 4: Delete lecture/Task/Exam

```
Begin
user chooses TimeTables -> Lectures/Tasks/Exams -> chooses a lecture/task/exam to be
deleted
if user chooses delete then delete lecture/task/exam from database
else do nothing
End
```

Algorithm 5: View details about a place

```
Begin
if user chooses Map then display interactive map
if user clicks on a place of interest then display details and pictures about that place
End
```

Algorithm 6: Add a new Note

```
Begin
user chooses Notes
if user clicks '+' sign and fills note details then save note in database
else do nothing
End
```

Algorithm 7: Calculate GPA

```
Begin
user chooses GPA Calculator
user enters credits and grade for each course
```

```
if user clicks CALCULATE then calculate GPA
else do nothing
End
```

5.4. Standards

Project Proposal Form: TUBITAK

Software Requirements Specification Document: IEEE 830-1998

Software Design Specification Document: IEEE 1016-1998

Security: Advanced Encryption Standard

Implementation: Google Development Guidelines[20]

5.5. Detailed description of the implementation

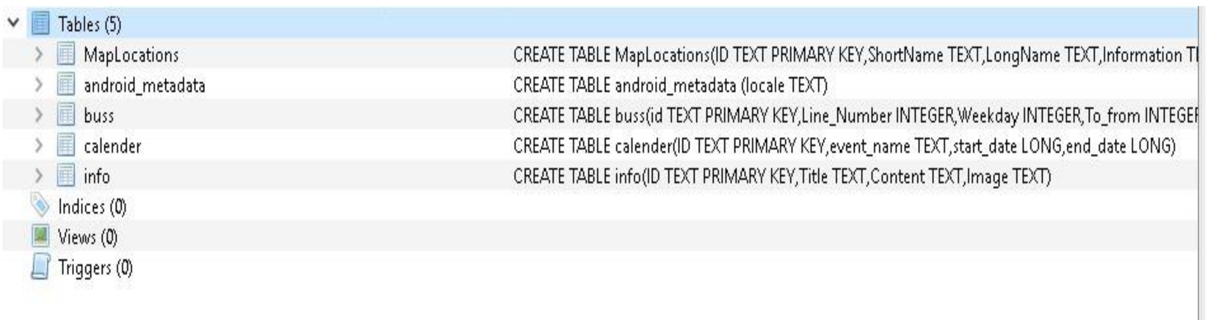
As the design and requirements step for each module is complete, the implementation step for that module begins. The implementation of the database is also considered in this Section. The application was developed using the Java Programming Language. There are a total of 22 activities in the implementation of the code on Android Studio. There are also a lot of layout xml files on Android Studio which work alongside the Java code.

There are two main databases that work with the application. One is the local database where all the application related data are stored and the other is an online hosted database which is constantly updated by the administrator. The local database file is called local.db and it is an SQL Database. This database was created using SQLite Library through Android. When the user runs the application, the app checks for the database file. If the database file was not found, a new empty database is constructed using the DB Constructor class. Then, a copy function is called to replace the empty database with an already filled database called firstdb.db which is included with the application folders. Another local database file called user.db is constructed when the application runs. This database is used to store the data that is entered by the user. Some data used by the app (eg. Map data) are stored as JSON files and are imported using GSON library. The implementations in Android studio along with the databases are shown in figures 26 to 34.

Data stored by local.db: MapLocations, Android_metadata, bus, calendar, info
Data stored by user.db: timetables, tasks, exams
Data stored by online database: Bus_Data, Calendar_Data, University_Locations

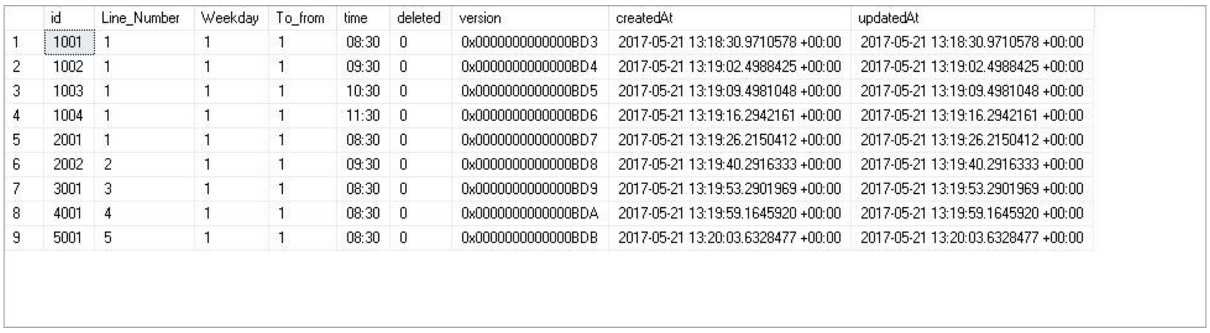
The purpose of the online database is to update the bus data, calendar data, and map data. The user cannot insert/modify any data on the online database. The online database is controlled by the administrator. The online database is hosted on Microsoft Azure Platform and managed using Microsoft SQL Managament Tool. When the user clicks on the update button in settings, the updater functions runs which starts by requesting data from the SQL server and if received, the updater deletes the data in the local database and inserts the newly obtained data.

The news module gets its data from EMU's RSS feed. There is function in the news module that downloads the news RSS xml file from EMU, and then it starts parsing the xml file and retrieves the information as a list of <item> classes. There is nothing that gets stored on the database from the news module. The user should be connected to the Internet to access them.



Tables (5)	
MapLocations	CREATE TABLE MapLocations(ID TEXT PRIMARY KEY,ShortName TEXT,LongName TEXT,Information T
android_metadata	CREATE TABLE android_metadata (locale TEXT)
buss	CREATE TABLE buss(id TEXT PRIMARY KEY,Line_Number INTEGER,Weekday INTEGER,To_from INTEGE
calender	CREATE TABLE calender(ID TEXT PRIMARY KEY,event_name TEXT,start_date LONG,end_date LONG)
info	CREATE TABLE info(ID TEXT PRIMARY KEY,Title TEXT,Content TEXT,Image TEXT)
Indices (0)	
Views (0)	
Triggers (0)	

Figure 26. Data stored by Local Database local.db



	id	Line_Number	Weekday	To_from	time	deleted	version	createdAt	updatedAt
1	1001	1	1	1	08:30	0	0x0000000000000BD3	2017-05-21 13:18:30.9710578 +00:00	2017-05-21 13:18:30.9710578 +00:00
2	1002	1	1	1	09:30	0	0x0000000000000BD4	2017-05-21 13:19:02.4988425 +00:00	2017-05-21 13:19:02.4988425 +00:00
3	1003	1	1	1	10:30	0	0x0000000000000BD5	2017-05-21 13:19:09.4981048 +00:00	2017-05-21 13:19:09.4981048 +00:00
4	1004	1	1	1	11:30	0	0x0000000000000BD6	2017-05-21 13:19:16.2942161 +00:00	2017-05-21 13:19:16.2942161 +00:00
5	2001	1	1	1	08:30	0	0x0000000000000BD7	2017-05-21 13:19:26.2150412 +00:00	2017-05-21 13:19:26.2150412 +00:00
6	2002	2	1	1	09:30	0	0x0000000000000BD8	2017-05-21 13:19:40.2916333 +00:00	2017-05-21 13:19:40.2916333 +00:00
7	3001	3	1	1	08:30	0	0x0000000000000BD9	2017-05-21 13:19:53.2901969 +00:00	2017-05-21 13:19:53.2901969 +00:00
8	4001	4	1	1	08:30	0	0x0000000000000BDA	2017-05-21 13:19:59.1645920 +00:00	2017-05-21 13:19:59.1645920 +00:00
9	5001	5	1	1	08:30	0	0x0000000000000BDB	2017-05-21 13:20:03.6328477 +00:00	2017-05-21 13:20:03.6328477 +00:00

Figure 27. The bus table in the local.db

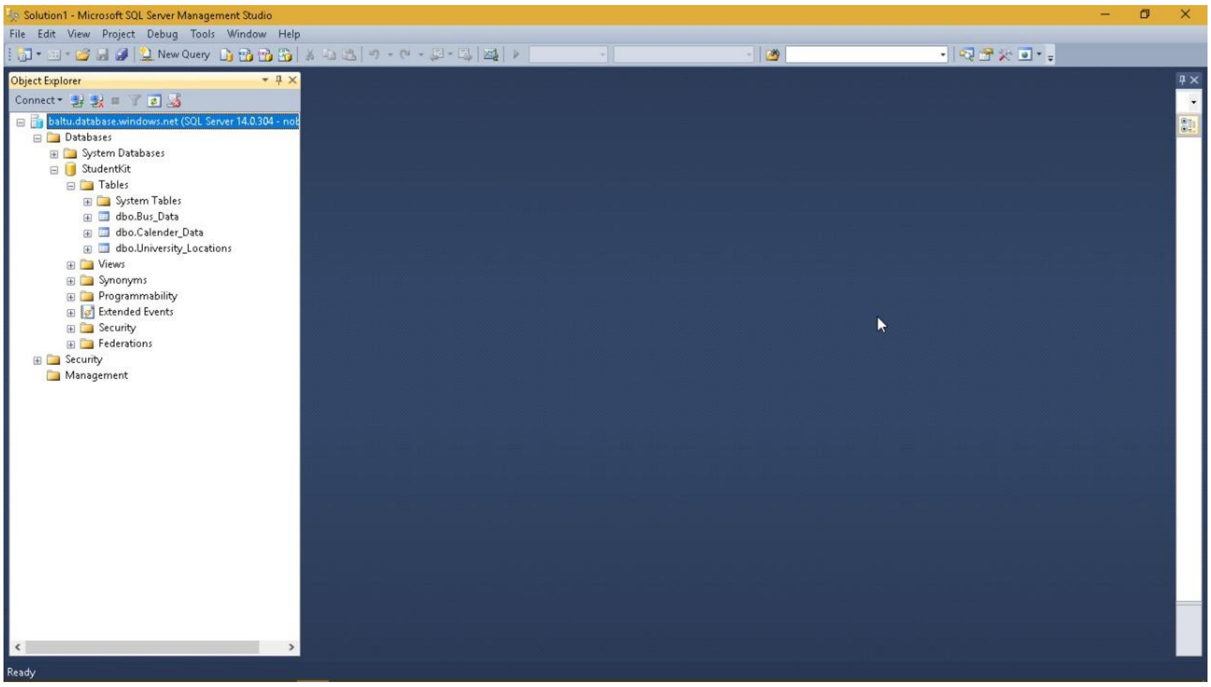
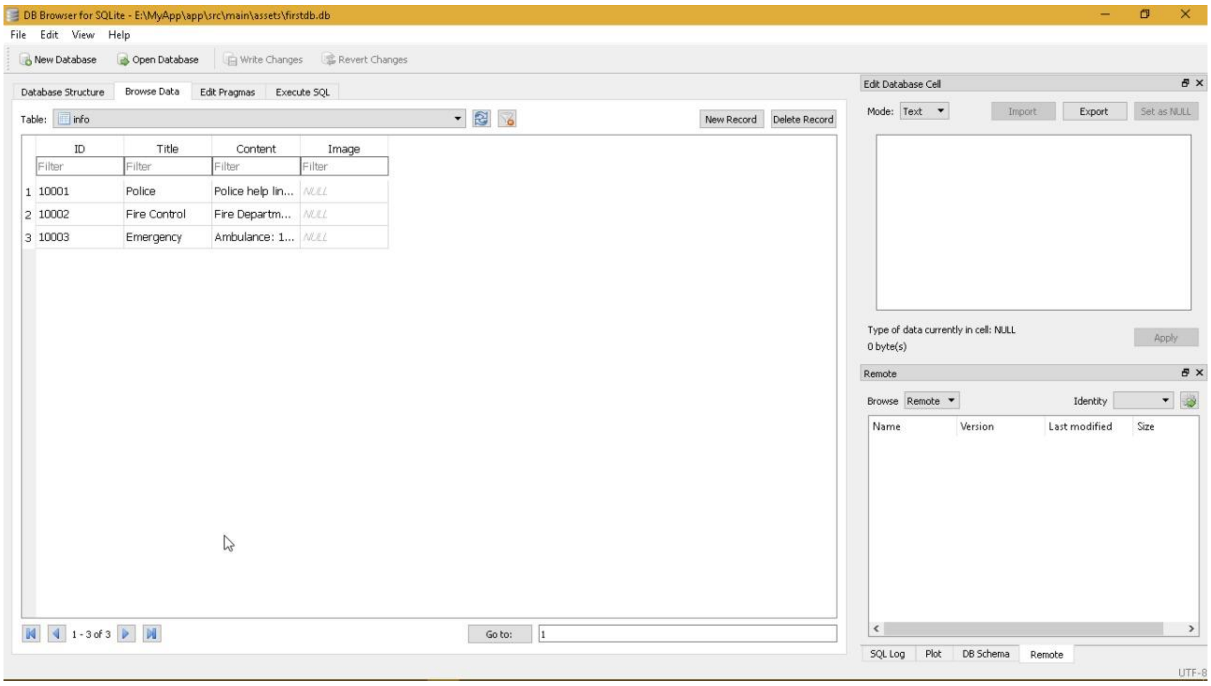


Figure 28. The online database on MS SQL Server Management Studio



	ID	Title	Content	Image
1	10001	Police	Police help lin...	NULL
2	10002	Fire Control	Fire Departm...	NULL
3	10003	Emergency	Ambulance: 1...	NULL

Figure 29. Info table from the DB Browser for SQLite

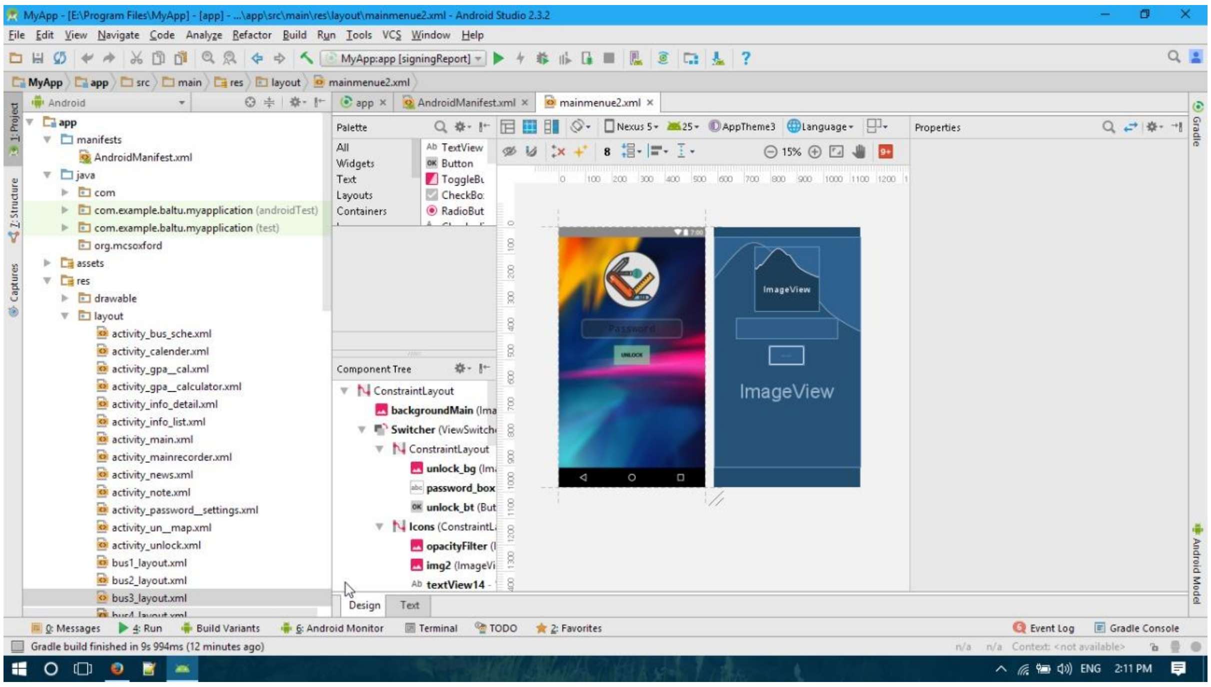
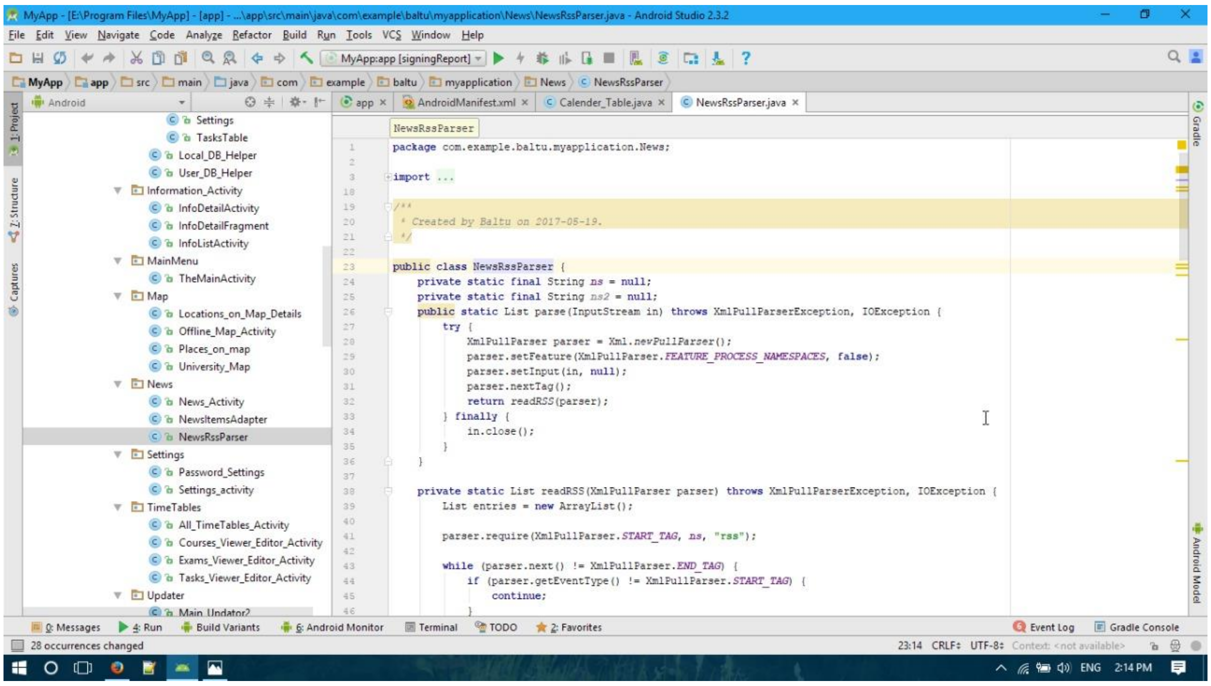
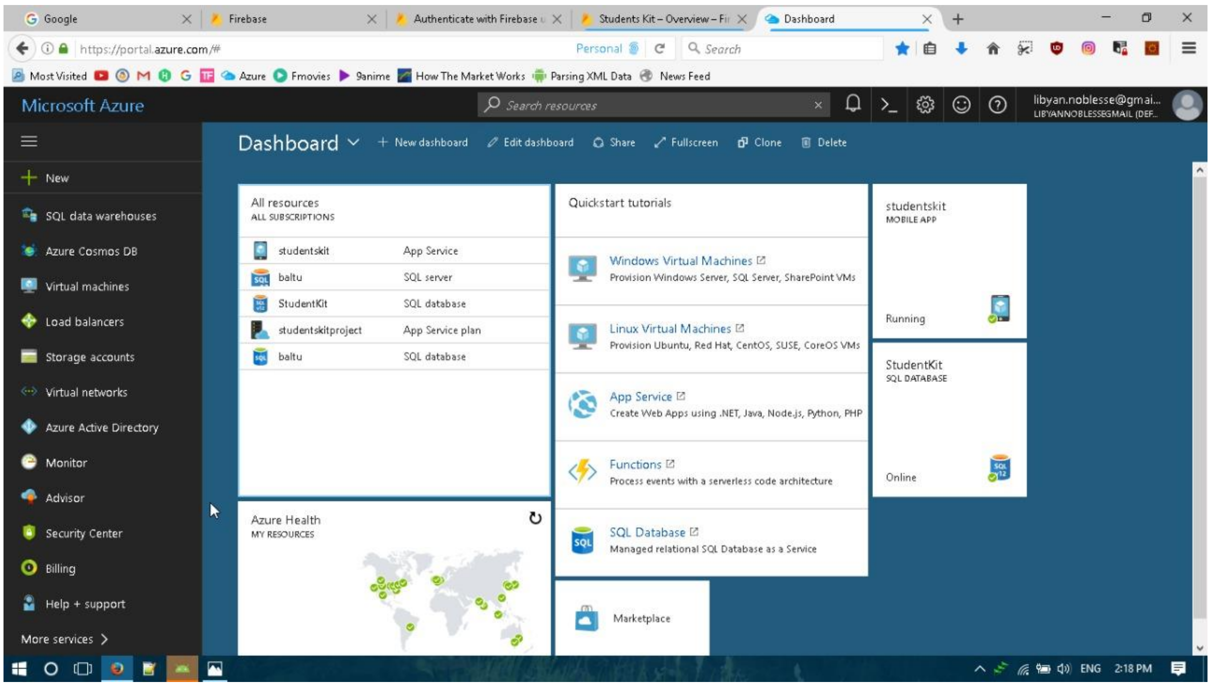


Figure 30. Implementation of Main using Android Studio


MyApp - [E:\Program Files\MyApp] - [app] - ...\app\src\main\java\com\example\baltu\myapplication\News\NewsRssParser.java - Android Studio 2.3.2
NewsRssParser.java
News_Activity
NewsItemsAdapter
NewsRssParser
28 occurrences changed

```java
package com.example.baltu.myapplication.News;

import ...

/**
 * Created by Baltu on 2017-05-19.
 */

public class NewsRssParser {
    private static final String ns = null;
    private static final String ns2 = null;
    public static List parse(InputStream in) throws XmlPullParserException, IOException {
        try {
            XmlPullParser parser = Xml.newPullParser();
            parser.setFeature(XmlPullParser.FEATURE_PROCESS_NAMESPACES, false);
            parser.setInput(in, null);
            parser.nextTag();
            return readRSS(parser);
        } finally {
            in.close();
        }
    }

    private static List readRSS(XmlPullParser parser) throws XmlPullParserException, IOException {
        List entries = new ArrayList();

        parser.require(XmlPullParser.START_TAG, ns, "rss");

        while (parser.next() != XmlPullParser.END_TAG) {
            if (parser.getEventType() != XmlPullParser.START_TAG) {
                continue;
            }
```

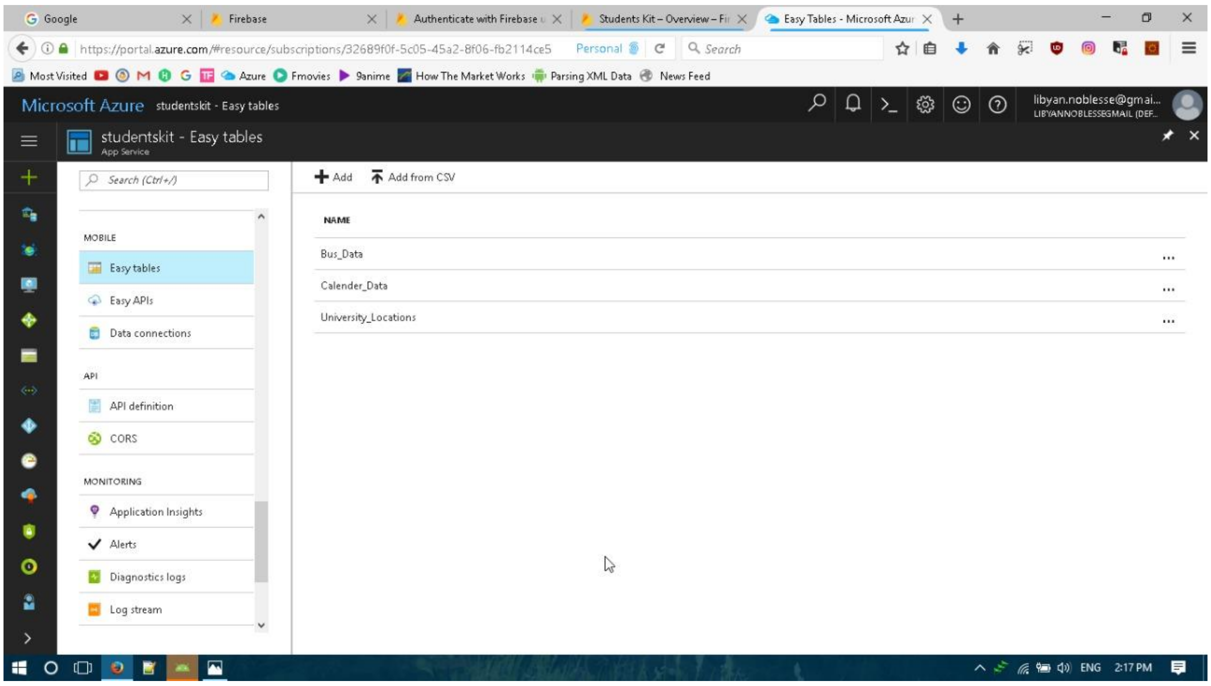
Figure 31. Implementation of the News Module using RSS feeds



All resources
ALL SUBSCRIPTIONS

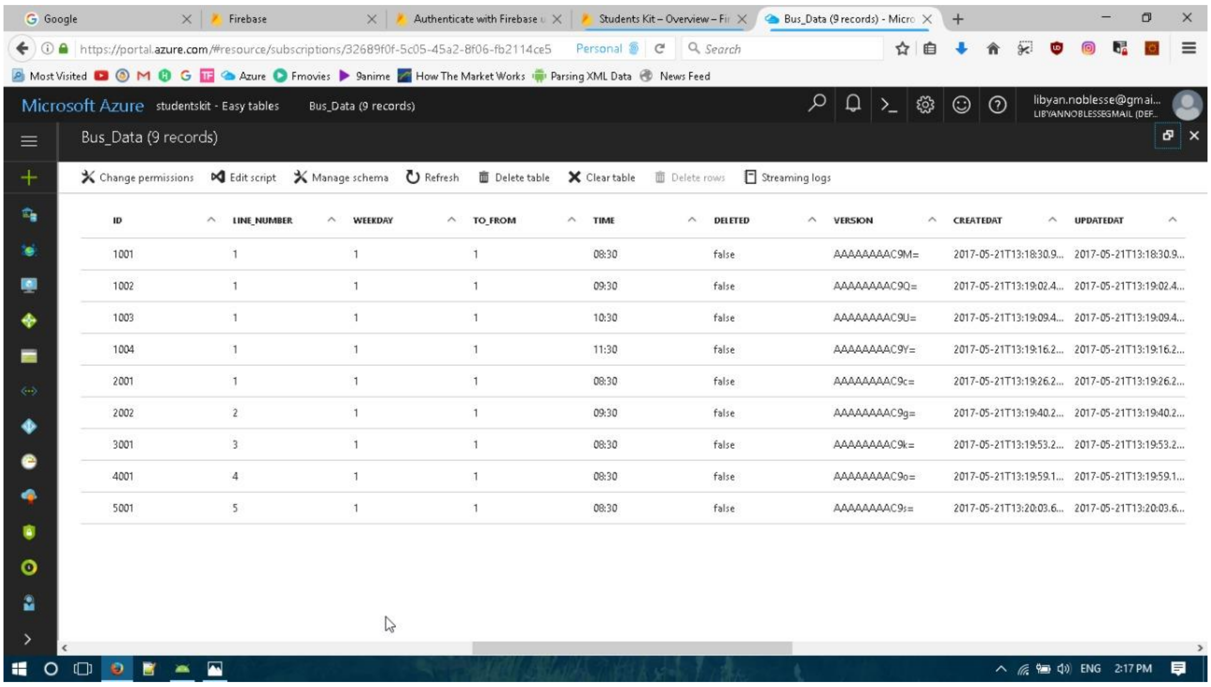
studentskit	App Service
baltu	SQL server
StudentKit	SQL database
studentskitproject	App Service plan
baltu	SQL database

Figure 32. SQL Server and Database on Microsoft Azure Platform



NAME
Bus_Data
Calender_Data
University_Locations

Figure 33. The online database tables (As in Fig. 26)



Bus_Data (9 records)

ID	LINE_NUMBER	WEEKDAY	TO_FROM	TIME	DELETED	VERSION	CREATEDAT	UPDATEDAT
1001	1	1	1	08:30	false	AAAAAAAAC9M=	2017-05-21T13:18:30.9...	2017-05-21T13:18:30.9...
1002	1	1	1	09:30	false	AAAAAAAAC9Q=	2017-05-21T13:19:02.4...	2017-05-21T13:19:02.4...
1003	1	1	1	10:30	false	AAAAAAAAC9U=	2017-05-21T13:19:09.4...	2017-05-21T13:19:09.4...
1004	1	1	1	11:30	false	AAAAAAAAC9Y=	2017-05-21T13:19:16.2...	2017-05-21T13:19:16.2...
2001	1	1	1	08:30	false	AAAAAAAAC9c=	2017-05-21T13:19:26.2...	2017-05-21T13:19:26.2...
2002	2	1	1	09:30	false	AAAAAAAAC9g=	2017-05-21T13:19:40.2...	2017-05-21T13:19:40.2...
3001	3	1	1	08:30	false	AAAAAAAAC9k=	2017-05-21T13:19:53.2...	2017-05-21T13:19:53.2...
4001	4	1	1	08:30	false	AAAAAAAAC9o=	2017-05-21T13:19:59.1...	2017-05-21T13:19:59.1...
5001	5	1	1	08:30	false	AAAAAAAAC9s=	2017-05-21T13:20:03.6...	2017-05-21T13:20:03.6...

Figure 34. The Bus_Data table on Microsoft Azure Platform

5. TESTING

Modules of the application were tested manually for any bugs, crashes as well as any unexpected output results. Expected outputs were compared with the outputs perceived during the testing and subsequent changes and modification to the code were made in case of mismatch. The validity of the end product has been verified by installing the software on several android devices with different specifications in order to assess visuals, resolution, colors and compatibility of the application. This method also allowed the use of these android devices to test the app during its development, not only to validate the end product. Testing results showed an overall good standing of the end product with major features of the application working smoothly without crashes or unexpected outputs. In addition, Android 4.3 and older versions showed no compatibility with the application. However, some minor bugs might be present that require further testing after the end product release due to the large number of variables that might affect the application's various features and functions.

Table 12. Test Case 1

Test Case ID	TC-01
Test Case Name	Login test
Pass/Fail Criteria	Pass: user enters correct password -> login user enters wrong password -> "wrong password"
Input Data	Numeric/Alphabetic/Symbolic Password
Test Procedure	Expected Output:
Step 1: Enter correct password	Logs in to system
Step 2: Enter wrong password	Displays a message "wrong password"
Comments	Test passes successfully

Table 13. Test Case 2

Test Case ID	TC-02
Test Case Name	Search/Notes Test
Pass/Fail Criteria	Pass: display notes that have title or description that matches the input text Fail: does not display a note/displays a different note
Input Data	text containing title or description of a note
Test Procedure	Expected Output:
Step 1: Create 3 notes containing the words "Design lab", "EMU", "Lab four" respectively	Creates 3 notes
Step 2: Search for the word "lab"	Displays the first and third word since they contain the word "lab"
Comments	Test passes successfully

Table 14. Test Case 3

Test Case ID	TC-03
Test Case Name	GPA Calculator Test
Pass/Fail Criteria	Pass: correct GPA output Fail: wrong GPA output
Input Data	Credits, grade for each course
Test Procedure	Expected Output:
Step 1: Course 1: Credits: 4, Grade: A	
Step 2: Course 2: Credits: 4, Grade: B	
Step 3: Course 2: Credits: 4, Grade: B	
Step 4: Course 2: Credits: 3, Grade: A	
Step 5: Course 2: Credits: 3, Grade: A	
Step 6: Calculate GPA	3.56
Comments	Test passes successfully

6. USER GUIDE OF THE SYSTEM

When the user launches the application, the menu shown in figure 35 will appear providing multiple features and options for the user to select. TimeTables allows the user to manage his courses, tasks and exams time tables. Calendar will display the academic year calendar of Eastern Mediterranean University. The map feature will display an interactive map of EMU campus. Buses icon will show to the user a list of buses lines the university provides with their respective schedules and times. News option allows the user to read EMU news, events, and announcements. Notes allow the user to create and edit notes for his academic needs. Info section will provide the user with general and helpful information about the Eastern Mediterranean University. GPA calculator when selected gives the user the opportunity to calculate his GPA as well as his CGPA. Finally, the Settings will allow the user to tweak different settings within the application to his liking. Figure 36 is what the user will see when selecting “TimeTables” from the main menu. Courses can be added to the TimeTable by selecting “add new” and days of the week can be scrolled through using the horizontal bar displaying the days of the week. In addition, pressing on the white rectangle on the top left of the application will display a drop down menu giving the option to switch to other time 2 tables, namely “Tasks” and “Exams” tables. Pressing on “Add new” will display a new interface shown in figure 37, here the user can input all the details needed to add his university course to the table.

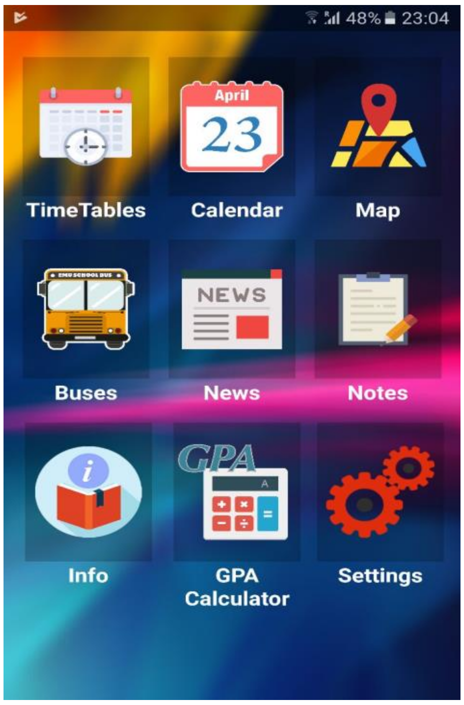


Figure 35. Main Menu of the application

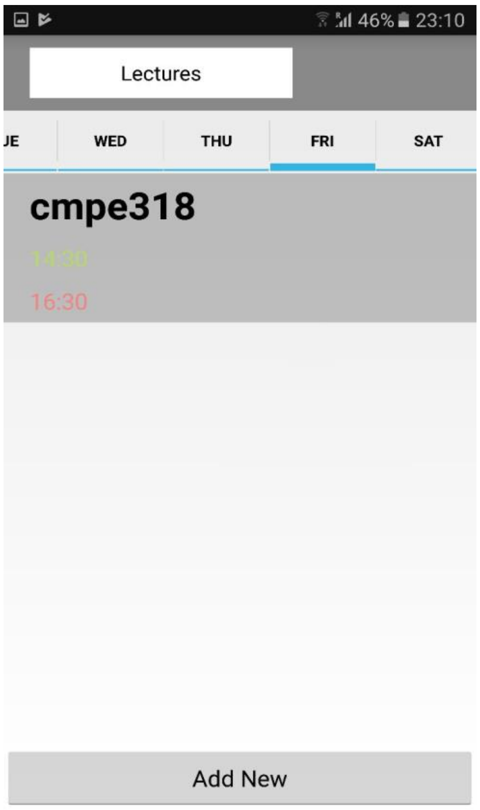


Figure 36. TimeTables Interface

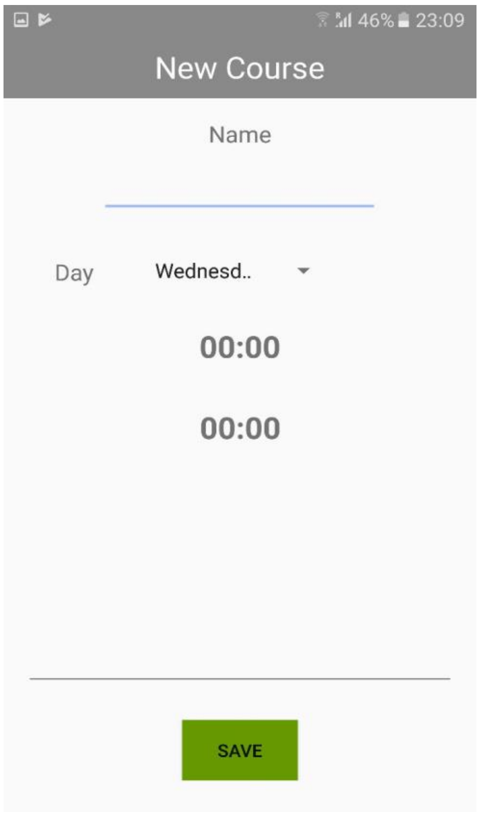
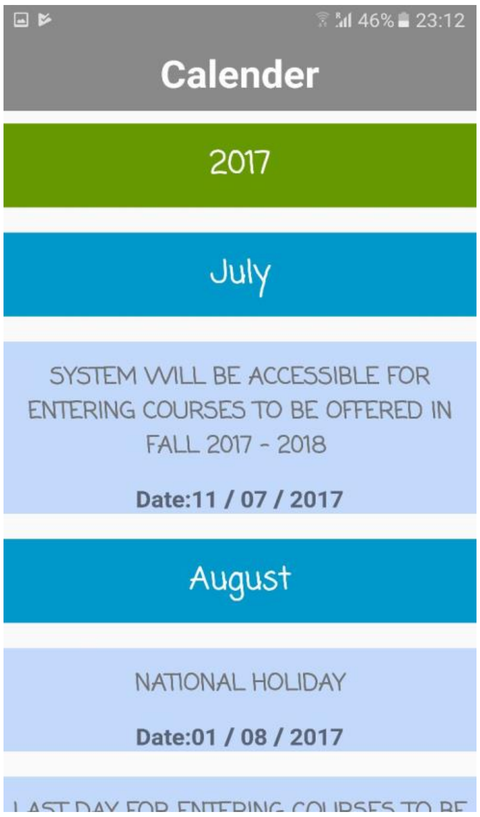


Figure 37. Add new course Interface


Calender

2017

July

SYSTEM WILL BE ACCESSIBLE FOR ENTERING COURSES TO BE OFFERED IN FALL 2017 - 2018

Date:11 / 07 / 2017

August

NATIONAL HOLIDAY

Date:01 / 08 / 2017

LAST DAY FOR ENTERING COURSES TO BE

Figure 38. Academic Calendar Interface

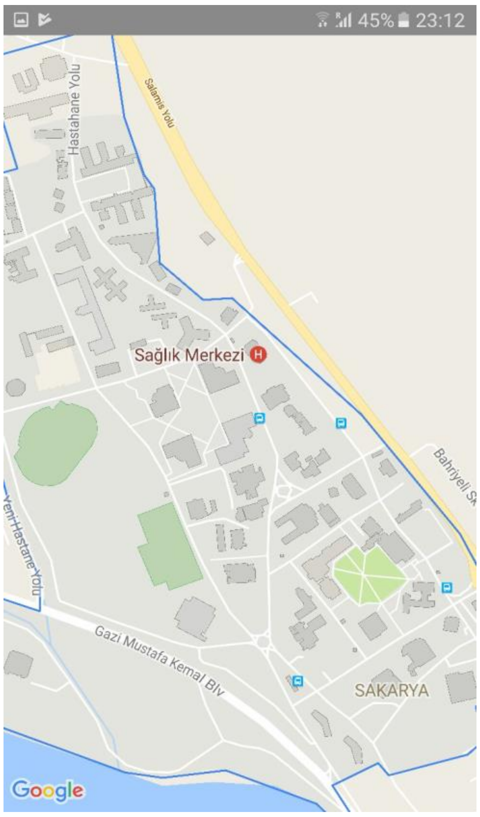
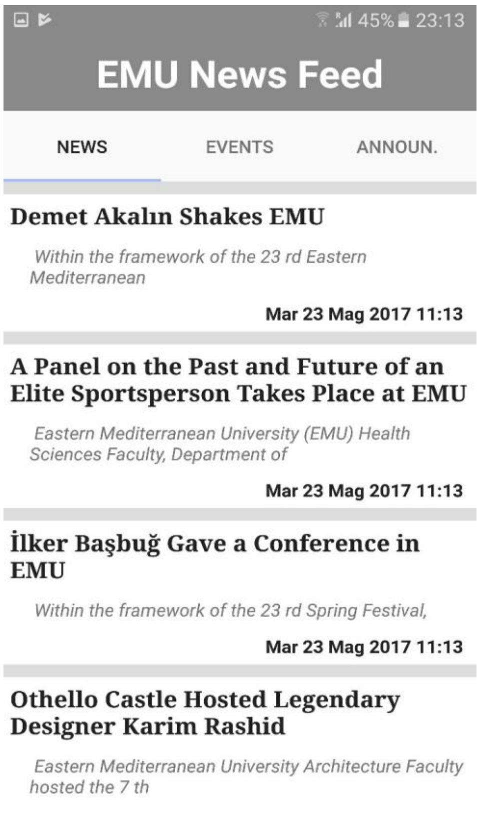


Figure 39. Map Interface


45% 23:13

EMU News Feed

NEWS EVENTS ANNOUN.

Demet Akalın Shakes EMU

Within the framework of the 23 rd Eastern Mediterranean

Mar 23 Mag 2017 11:13

A Panel on the Past and Future of an Elite Sportsperson Takes Place at EMU

Eastern Mediterranean University (EMU) Health Sciences Faculty, Department of

Mar 23 Mag 2017 11:13

İlker Başbuğ Gave a Conference in EMU

Within the framework of the 23 rd Spring Festival,

Mar 23 Mag 2017 11:13

Othello Castle Hosted Legendary Designer Karim Rashid

Eastern Mediterranean University Architecture Faculty hosted the 7 th

Figure 40. News Interface

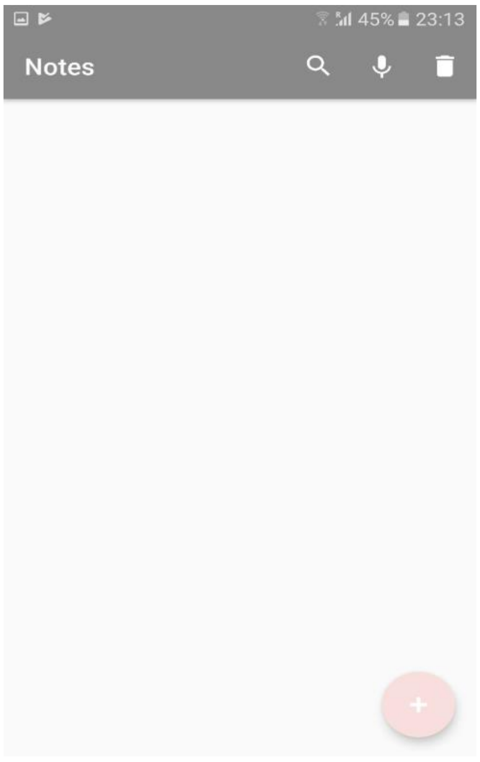


Figure 41. Notes Interface

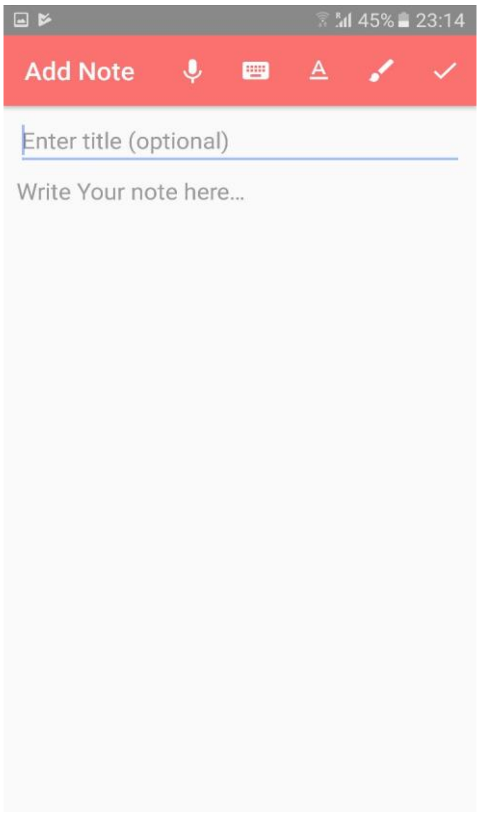
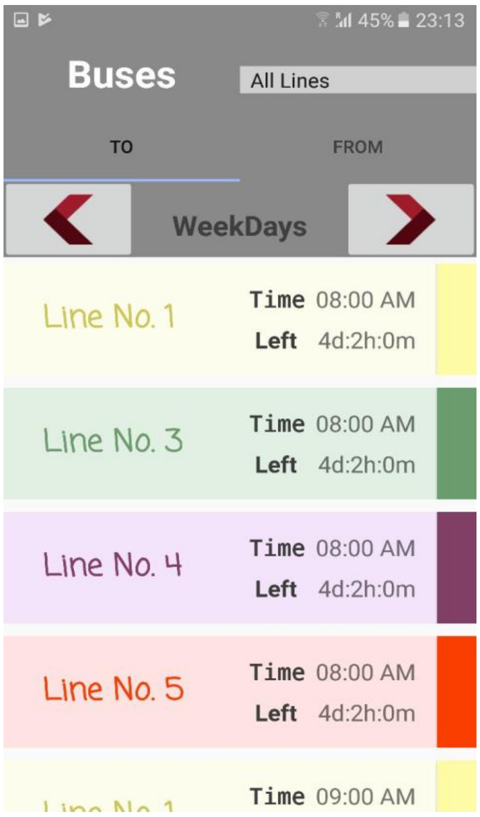


Figure 42. Add Note Interface

The notes feature allows the user to create a new note, search for an existing note, delete existing notes as well as record notes in form of an audio file as shown in Figure 41. Figure 42 is the interface directed to when the user presses the "+" button on the bottom right of the screen as shown in Figure 41. When adding a new note, a user can use several features such as using voice to write down notes, changing color and making the text bold, cursive and underlined. The feature to draw in the note is also available. Once saved, a note can be edited later on. Figure 47 shows the interface of the Settings icon where the user can tweak several settings, General allows the user to set up a password in order to lock the app for future uses with a password, Pressing on update will update the application and the features within it. Help button displays helpful information and guidelines regarding the application. The about button provides the user with contact and general information about the application developers and the team involved.


Buses

All Lines

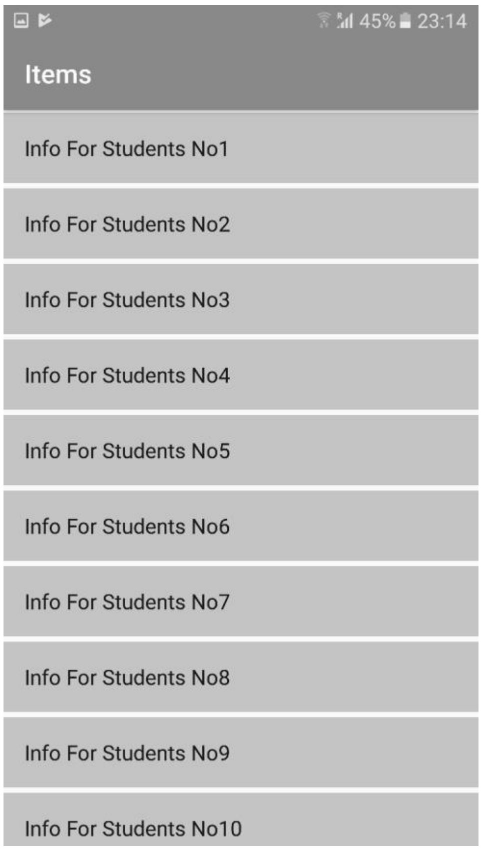
TO

FROM

WeekDays

Line No. 1	Time	08:00 AM
	Left	4d:2h:0m
Line No. 3	Time	08:00 AM
	Left	4d:2h:0m
Line No. 4	Time	08:00 AM
	Left	4d:2h:0m
Line No. 5	Time	08:00 AM
	Left	4d:2h:0m
Line No. 1	Time	09:00 AM

Figure 43. EMU Buses lines & schedules


Items

Info For Students No1

Info For Students No2

Info For Students No3

Info For Students No4

Info For Students No5

Info For Students No6

Info For Students No7

Info For Students No8

Info For Students No9

Info For Students No10

Figure 44. sample slots in the info section

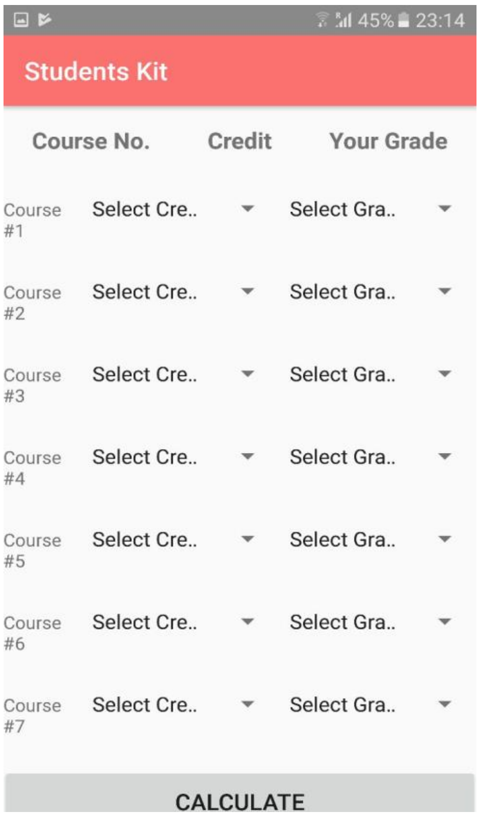


Figure 45. GPA and CGPA calculator interface continuation

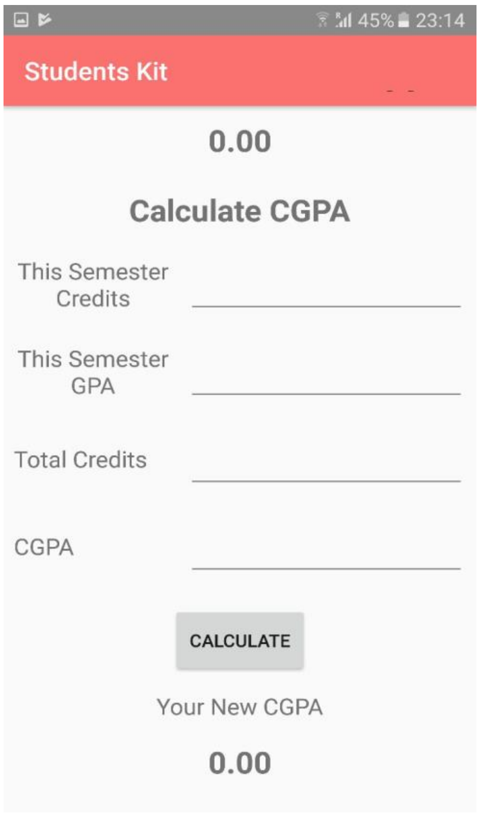


Figure 46. GPA and CGPA calculator

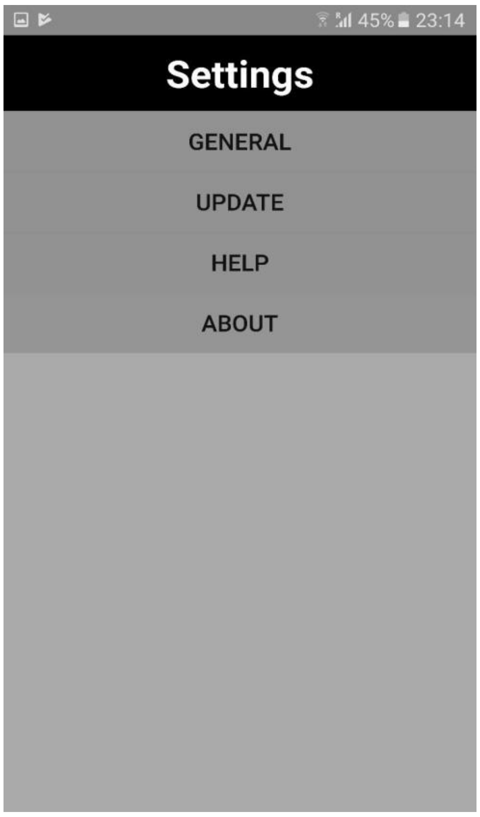


Figure 47. Settings interface

7. DISCUSSION

The EMU Student Kit can be of great relief to EMU Students and to other students. EMU Student Kit is a not-for-profit and an open source project which has a good probability for increasing the performance of EMU Students by assisting them in increasing their performance in various fields such as time management, organization and productivity. The time tables section of the application is the most important section where the user can organize all his courses, exams, and tasks in an easy to use manner. The bus schedules keeps the students updated on all routes and timings that the buses of EMU take. This is so useful especially since the timings of the buses can change at random times and during exam times. The Interactive map along with the bus schedules can be of great value, especially to newly registered students. The interactive map allows the student to view information about any available point of interest inside EMU Campus. The info section can be a real life saver since it contains all the important numbers and information about EMU and North Cyprus like emergency numbers, fire department, police department, psychological aid, animal control, etc. The notes section can keep the student organized and productive since it not only supports notes, but also supports Audio Recording and Drawing which can be very useful during lectures. The news section efficiently keeps the student updated on all the news/events/announcements of EMU by providing an easy and quickly navigable interface instead of actually visiting EMU website and clicking on slow to load links. The student can also access the academic calendar quickly instead of searching for it on EMU Website. Students often do not know how to calculate their GPA/CGPA or are too lazy to do it. The GPA/CGPA calculator eliminates those things by providing an efficient and easy to use interface for calculating a student’s GPA/CGPA. The password feature is a very important feature included in the application. It allows the student to very securely store any sensitive data which can be entered in the application since the password is equipped with the Advanced Encryption Standard which is one of the best standards for encryption. Therefore, EMU Student Kit can be of great benefit to the student of EMU. The unique thing about this application is that it acts as a complete student package and provides everything the student of EMU needs to succeed.

9. CONCLUSION

The EMU Student Kit is an educational Android mobile application made for EMU Students. The application can act as an ultimate guide to EMU and can be really useful in keeping the students productive and organized. It is a great application considering it has been developed in only 4 months which is a relatively short time. The EMU Student Kit has the potential to grow further and further as new features gets added in the future and I predict a bright successful future for the application. EMU Student Kit can also act as an example to other organizations on how a properly designed system acts like and other organizations could benefit from the application's designs and functionalities in producing their own applications. Software Engineering students may also learn a lot of things by referring to this application and the development of this application helped me personally by giving me the chance to learn the Java programming language along with many crucial software engineering activities. In the end, we are going to publish this application to Google's Play Store to make it easier for everyone to install it. We are also planning to release many new features in the future such as integration with EMU's Student Portal.

10. REFERENCES

[1]. IEEE Recommended Practice for Software Requirements Specifications. (1998). Ieeexplore.ieee.org. Retrieved 27 May 2017, from http://ieeexplore.ieee.org/document/720574/

[2]. IEEE Recommended Practice for Software Design Descriptions. (1998). Ieeexplore.ieee.org. Retrieved 27 May 2017, from http://ieeexplore.ieee.org/document/741934/

[3]. Project Management Software | Microsoft Project. (2016). Products.office.com. Retrieved 27 May 2017, from https://products.office.com/en-us/project/project-and-portfolio-management-software?tab=tabs-1

[4]. Software Design Tools for Agile Teams, with UML, BPMN and More. Visual-paradigm.com. Retrieved 27 May 2017, from https://www.visual-paradigm.com/

[5]. Flowchart Maker & Diagramming Software, Microsoft Visio. (2016). Pr oducts.office.com. Retrieved 27 May 2017, from https://products.office.com/en-us/visio/flowchart-software?tab=tabs-1

[6]. MockFlow - Online Wireframe and UX Tools. Mockflow.com. Retrieved 27 May 2017, from https://www.mockflow.com/

[7]. Gliffy | Online Diagram and Flowchart Software. Gliffy.com. Retrieved 27 May 2017, from https://www.gliffy.com/

[8]. GenMyModel. GenMyModel. Retrieved 27 May 2017, from https://www.genmymodel.com/

[9]. Flowchart Maker & Online Diagram Software. Draw.io. Retrieved 27 May 2017, from http://draw.io

[10]. SQL Server 2016 | Microsoft. Microsoft SQL Server - US (English). Retrieved 27 May 2017, from https://www.microsoft.com/en-us/sql-server/sql-server-2016

[11]. Download Android Studio and SDK Tools | Android Studio. Developer.android.com. Retrieved 27 May 2017, from https://developer.android.com/studio/index.html

[12]. Microsoft Azure: Cloud Computing Platform & Services. Azure.microsoft.com. Retrieved 27 May 2017, from http://azure.microsoft.com

[13]. SQLite Home Page. Sqlite.org. Retrieved 27 May 2017, from https://www.sqlite.org/

[14]. JSON. Json.org. Retrieved 27 May 2017, from http://www.json.org/

[15]. Notepad++ Home. Notepad-plus-plus.org. Retrieved 27 May 2017, from https://notepad-plus-plus.org/

[16]. DB Browser for SQLite. Sqlitebrowser.org. Retrieved 27 May 2017, from http://sqlitebrowser.org/

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

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

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

[20]. Android Developers. Developer.android.com. Retrieved 27 May 2017, from

APPENDIX A

Instructions for Installing the system:

Copy the file “student-kit.apk” on the CD accompanied with this report to any Android device with an Android Version of 4.4 or higher. On the Android device, locate the folder in which the file was copied, open the .apk file and click install.

APPENDIX B

Code for the system: TheMainActivity.java

This is the code for the main activity of the program. The code for all the other parts of the program can be found on the CD attached with the report.

```
package com.example.baltu.myapplication.MainMenu;

import android.app.Activity;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteException;

import android.database.sqlite.SQLiteOpenHelper;

import android.graphics.Point;

import android.net.ConnectivityManager;

import android.net.NetworkInfo;

import android.os.Bundle;

import android.util.Log;

import android.view.Display;

import android.view.View;

import android.view.WindowManager;

import android.view.inputmethod.InputMethodManager;

import android.widget.EditText;

import android.widget.Toast;

import android.widget.ViewSwitcher;

import com.example.baltu.myapplication.Buses_Schedules;

import com.example.baltu.myapplication.Academic_Calendar;

import com.example.baltu.myapplication.DB_Constructor.Local_DB_Helper;

import com.example.baltu.myapplication.DB_Constructor.User_DB_Helper;

import com.example.baltu.myapplication.GPA_Calculator;

import com.example.baltu.myapplication.Information_Activity.InfoListActivity;

import com.example.baltu.myapplication.News.News_Activity;

import com.example.baltu.myapplication.R;

import com.example.baltu.myapplication.Settings.Settings_activity;
```

```
;

import com.example.baltu.myapplication.Map.University_Map;
import com.example.baltu.myapplication.Map.Offline_Map_Activity;
import com.example.baltu.myapplication.TimeTables.All_TimeTables_Activity;
import com.example.tomek.notepad.MainActivity;
import com.scottyab.aescrypt.AESCrypt;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.security.GeneralSecurityException;
/*Unfortunately The code was rushed and implemented by only one student
* so the focus was on the functionality and no focus on the documentation*/
public class TheMainActivity extends Activity {
    Point sizes = new Point();
    SQLiteDatabase userdatabase;
    SQLiteDatabase localdatabase;
    Integer Tries = 3;
    static String DB_PATH = "/data/data/com.example.baltu.myapplication/databases/";
    public static final String DB_FILENAME = "LocalDb.db";
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.mainmenue2);
        findViewById(R.id.img3).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intents2 = new Intent(TheMainActivity.this, Academic_Calendar.class);
                startActivity(intents2);
            }
        });
        findViewById(R.id.img5).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent calenderintents = new Intent(TheMainActivity.this, Buses_Schedules.class);
                startActivity(calenderintents);
            }
        });
        findViewById(R.id.img6).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (isNetworkAvailable()) {
                    Intent NewsIntent = new Intent(TheMainActivity.this, News_Activity.class);//News
                    startActivity(NewsIntent);
                } else
                    Toast.makeText(TheMainActivity.this, "Not Connected To the Internet",
Toast.LENGTH_SHORT).show();
            }
        });
        findViewById(R.id.img7).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent NotesIntent = new Intent(TheMainActivity.this, MainActivity.class);//Notes
                startActivity(NotesIntent);
            }
        });
        findViewById(R.id.img8).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent InformationIntent = new Intent(TheMainActivity.this, InfoListActivity.class);
                startActivity(InformationIntent);
            }
        });
        findViewById(R.id.img11).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent SettingsIntent = new Intent(TheMainActivity.this, Settings_activity.class);
                startActivity(SettingsIntent);
            }
        });
        findViewById(R.id.img2).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent TimetablesIntent = new Intent(TheMainActivity.this, All_TimeTables_Activity.class);
                startActivity(TimetablesIntent);
            }
        });
        findViewById(R.id.img9).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent gpaIntent = new Intent(TheMainActivity.this, GPA_Calculator.class);
                startActivity(gpaIntent);
            }
        });
        WindowManager wm = (WindowManager) this.getSystemService(Context.WINDOW_SERVICE);
        Display display = wm.getDefaultDisplay();
        display.getSize(sizes);
        Log.d("MainMenue", "onCreate");
        final SharedPreferences prefs = getSharedPreferences(getString(R.string.preference_file_key),
MODE_PRIVATE);
        boolean key = prefs.getBoolean("pass_on", false);
        final String password_string = prefs.getString("Password_String", "");
        final ViewSwitcher vs = (ViewSwitcher) findViewById(R.id.Switcher);
        if (!key) {
            Log.d("Mainmen", "no pass");
            vs.showNext();
            if (checkDataBase()) {
                SQLiteOpenHelper openhelper = new User_DB_Helper(this);
                userdatabase = openhelper.getWritableDatabase();
                userdatabase.close();
                SQLiteOpenHelper localdbhelper = new Local_DB_Helper(this);
                localdatabase = localdbhelper.getWritableDatabase();
                localdatabase.close();
                Toast.makeText(this, "not coppied", Toast.LENGTH_SHORT).show();
            } else {
                SQLiteOpenHelper openhelper = new User_DB_Helper(this);
                userdatabase = openhelper.getWritableDatabase();
                userdatabase.close();
                SQLiteOpenHelper localdbhelper = new Local_DB_Helper(this);
                localdatabase = localdbhelper.getWritableDatabase();
                localdatabase.close();
                try {
                    copyDataBase();
                    Toast.makeText(this, "coppied", Toast.LENGTH_SHORT).show();
                } catch (IOException e) {
                    e.printStackTrace();
                    Toast.makeText(this, "Error", Toast.LENGTH_SHORT).show();
                }
            }
            //Toast.makeText(this, "Created DB", Toast.LENGTH_SHORT).show();
        } else {
            findViewById(R.id.unlock_bt).setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    EditText pass = (EditText) findViewById(R.id.password_box);
                    if (!pass.getText().toString().equals("")) {
                        try {
                            if (AESCrypt.encrypt("CanTheWorldUnderstand?", "TuNiSbYnIgHt" +
pass.getText().toString()).equals(password_string)) {
                            SharedPreferences.Editor PrefE = prefs.edit();
                            PrefE.putBoolean("Logged", true);
                            PrefE.commit();
                            Log.i("logged", "true");
                            View view = TheMainActivity.this.getCurrentFocus();
                            if (view != null) {
                                InputMethodManager imm = (InputMethodManager) getSystemService(Context.INPUT_METHOD_SERVICE);
                                imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
                            }
                            pass.setText("");
                            vs.showNext();
                            } else {
                            View view = TheMainActivity.this.getCurrentFocus();
                            if (view != null) {
                                InputMethodManager imm = (InputMethodManager) getSystemService(Context.INPUT_METHOD_SERVICE);
                                imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
                            }
                            Tries--;
                            if (Tries == 0)
                                finish();
                            else {
                                Toast.makeText(TheMainActivity.this, "    Wrong Password" + "\nYou Have " + Integer.toString(Tries) + " More Tries", Toast.LENGTH_SHORT).show();
                                pass.setText("");
                            }
                            }
                        } catch (GeneralSecurityException e) {
                            e.printStackTrace();
                        }
                    } else
                        Toast.makeText(TheMainActivity.this, "Enter Password", Toast.LENGTH_SHORT).show();
                }
            });
        }
      }
```

```
@Override
protected void onStart() {
    super.onStart();
    EditText pass = (EditText) findViewById(R.id.password_box);
    pass.setText("");
}
public void clickonMap(View view) {
    if (isNetworkAvailable()) {
        Intent intf = new Intent(this, University_Map.class);
        intf.putExtra("type", 0);
        startActivity(intf);
    } else {
        Toast.makeText(this, "connect to the internet for an interactive map", Toast.LENGTH_LONG).show();
        Intent offlineMap = new Intent(TheMainActivity.this, Offline_Map_Activity.class);
        startActivity(offlineMap);
    }
}
private boolean isNetworkAvailable() {
    ConnectivityManager connectivityManager
            = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo();
    return activeNetworkInfo != null && activeNetworkInfo.isConnected();
}
private void copyDataBase() throws IOException {
    try {
        InputStream mInputStream = this.getAssets().open("firstdb.db");
        String outFileName = DB_PATH + DB_FILENAME;
        OutputStream mOutputStream = new FileOutputStream(outFileName);
        byte[] buffer = new byte[1024];
        int length;
        while ((length = mInputStream.read(buffer)) > 0) {
            mOutputStream.write(buffer, 0, length);
            }
            mOutputStream.flush();
            mOutputStream.close();
            mInputStream.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    private boolean checkDataBase() {
        try {
            final String mPath = DB_PATH + DB_FILENAME;
            final File file = new File(mPath);
            if (file.exists())
                return true;
            else
                return false;
        } catch (SQLiteException e) {
            e.printStackTrace();
            return false;
        }
    }
}
```

APPENDIX C

Abbreviations and Definitions:

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
IDE:	Integrated Development Environment
SQL:	Structured Query Language
MAU:	Monthly Average Users
OS:	Operating System
GPA:	Grade Point Average
CGPA:	Cumulative Grade Point Average
XML:	Extensible Markup Language
RSS:	Rich Site Summary
Activity:	Java code that supports a screen or UI, i.e., building block of the user interface is an activity.