Team Project SRS Team #8

Spring 2018

Alexandre Allani, Shin Seung Hun, Kuntae Park, Bilgehan Bingol

April 24, 2018

This document is the Software Requirements Specification of our project.

Contents

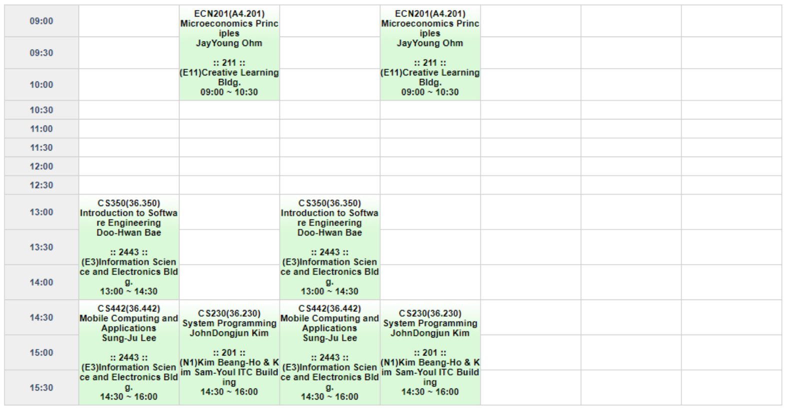
1 Introduction

The product presented in this SRS document is an Android mobile application which will run on smartphone devices. The basis of this application is to provide its user a dynamic timetable on his smartphone. The software is aimed to help all KAIST users in managing their schedule.

This application might be very handy, especially if you're not a fan of the portal KAIST timetable. Indeed, even though the timetable is accessible through the portal, it is not designed for mobile devices such as smartphones. Moreover, this timetable is fixed, meaning we can't see last-minutes updates. If a teacher has to change the room, you can only find this information on your emails. This is where this application is also different from Google Agenda or any kind of agenda-typed application.

Top level Requirement

Overall, our schedule application will have many functions that will satisfy user's needs. First of all, it will provide the basic and expected schedule you can find on https://ssogw6.kaist.ac.kr /timeTable. However, as it is possible to see in **Figure 1**, this only provides information about official classes. Also, the timetable is time-frozen, only the usual week is displayed. Consequently, the application will display not only class-related schedule but also different events that will happen in KAIST. Moreover, it will update any information related to a KAIST system. This means that if a professor add some lab sessions or extra classes, it will be automatically updated on the application
In the same way, the user will be able to schedule his studies session in KAIST and put it in the timetable according to the other events. Furthermore, a user can also share events among KAIST students. For instance, if one user wants to make a study group session, it will add everyone in the event. Finally, the user will receive notification of an upcoming event.
To sum up, the main requirement is to display a dynamic and connected schedule to KAIST students.



09:00		ECN201(A4.201) Microeconomics Princ iples JayYoung Ohm :: 211 :: (E11)Creative Learning Bldg. 09:00 ~ 10:30		ECN201(A4.201) Microeconomics Princ iples JayYoung Ohm :: 211 :: (E11)Creative Learning Bldg. 09:00 ~ 10:30			
09:30							
10:00							
10:30							
11:00							
11:30							
12:00							
12:30							
13:00	CS350(36.350) Introduction to Softwa re Engineering Doo-Hwan Bae :: 2443 :: (E3)Information Scien ce and Electronics Bld g. 13:00 ~ 14:30		CS350(36.350) Introduction to Softwa re Engineering Doo-Hwan Bae :: 2443 :: (E3)Information Scien ce and Electronics Bld g. 13:00 ~ 14:30				
13:30							
14:00							
14:30	CS442(36.442) Mobile Computing and Applications Sung-Ju Lee :: 2443 :: (E3)Information Scien ce and Electronics Bld g. 14:30 ~ 16:00	CS230(36.230) System Programming JohnDongjun Kim :: 201 :: (N1)Kim Beang-Ho & K im Sam-Youl ITC Build ing 14:30 ~ 16:00	CS442(36.442) Mobile Computing and Applications Sung-Ju Lee :: 2443 :: (E3)Information Scien ce and Electronics Bld g. 14:30 ~ 16:00	CS230(36.230) System Programming JohnDongjun Kim :: 201 :: (N1)Kim Beang-Ho & K im Sam-Youl ITC Build ing 14:30 ~ 16:00			
15:00							
15:30							

Figure 1: Kaist Portal TimeTable

2 Overall Perspective

In this part, we'll try to describe the perspective of this project and how we are thinking it up in the future.

2.1 Product Perspective

This product is aimed to be a further use of the timetable on the website. Since this product combines both a typed-agenda application and KAIST timetable, we can imagine that it can also be considered as a replacement of the existing timetable available on KAIST website.
The system is composed of four main components:

- GUI: Graphical user interface. This component allows users to interact with the system
- KAIST database: contains all the scheduling information relative to the class and studies
- Teaching interface: Teacher updates database from this interface
- Data control algorithm: Handles all the data received from the database, and update the GUI. Modifies also the GUI if a request was made by the user (for instance if the user want to add something to its schedule, a user interacts with GUI which sends the information to the Data Control algorithm which then will update the GUI)

The diagram below explains the interactions between each component:

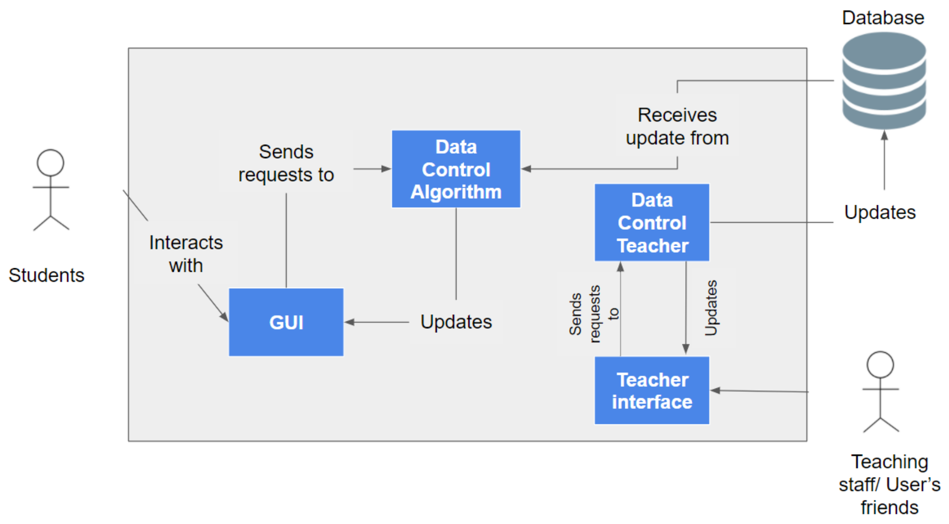


Figure 2: Component Diagram

Using the components shown above, we will develop a timetable application specific to KAIST users first. The purpose here is to provide students a dynamic and customized timetable they can modify according to their own schedule.

2.2 Product Features

The product rest upon the four main features listed below:

- Display a Graphical schedule according to user's classes and preferences.
- Modify the schedule according to user's own schedule
- Create an event that other users can add to their schedules (e.g. study group sessions, baseball match ...)
- Update automatically schedules in case of changes from teacher staff (e.g. change of class hours, extra classes) or other students (e.g. change of baseball match's hours).

This software is unique because it is adapted to KAIST students. Indeed, as mentioned before, we could compare this application to Google Agenda, or the existing Timetable on KAIST portal. Why is it original then? Our service proposes better interactions between KAIST students. For instance, if someone wants to play volleyball with other students, he can add this event to its schedule and invite other friends to add it. This feature does not exist on Google agenda. Moreover, since this application is internal to KAIST, there is no need to worry about data protection.

2.3 Operating Environment

In order to run our software, operating environment is expected as follow:

- Operating System: Android Oreo (8.1) with application compatibility until Android KitKat (4.4). Since we want most users to be able to run our application, choosing this version will include 94.8% of Android users (see Appendix).
- Devices: It will be adapted to smartphones, probably not on tablets. We have chosen to make it optimized on smartphones because, as for the choice of operating system, there are far more people (90.8%) using smartphones than tablets (see Appendix).
- Network requirements: The software will need an Internet connection in order to communicate with the database and update schedules. However, the system can also work offline.
- The database should be installed in KAIST facilities. We don't know if KAIST already has a database with an API allowing us to retrieve all data needed. In the rest of the SRS, we will assume that we have a database to which each user can connect. Moreover, this database will have an API. Consequently, we can get data (probably JSON objects) from the database and post data on it.

2.4 Assumptions and Dependencies

Here is a list of different assumptions we are making:

- **Database assumptions.** As mentioned before, some assumptions should be made on the Database. During the development of the software, we will suppose that our application will receive JSON objects from a distant database. The assumption here is that we suppose, such database already exists. However, for the purpose of this project, our database will be a file stored on a computer, and requests will be made manually.
- A user won't have to sign up. He will use the same IDs he uses on KAIST portal and KLMS.
- We assume that there are no external attacks on the servers nor application. However, privacy being a significant part of our application, in further releases, this assumption will be false.
- We assume that students use only Android phones. Even though most people use Android, the application should be available on IOS since it is aimed to help all students.
- We assume that after the first release, the software won't require maintenance.
- We assume that users won't abuse the system. Changing other's schedule by exploiting a loophole.

Dependencies list :

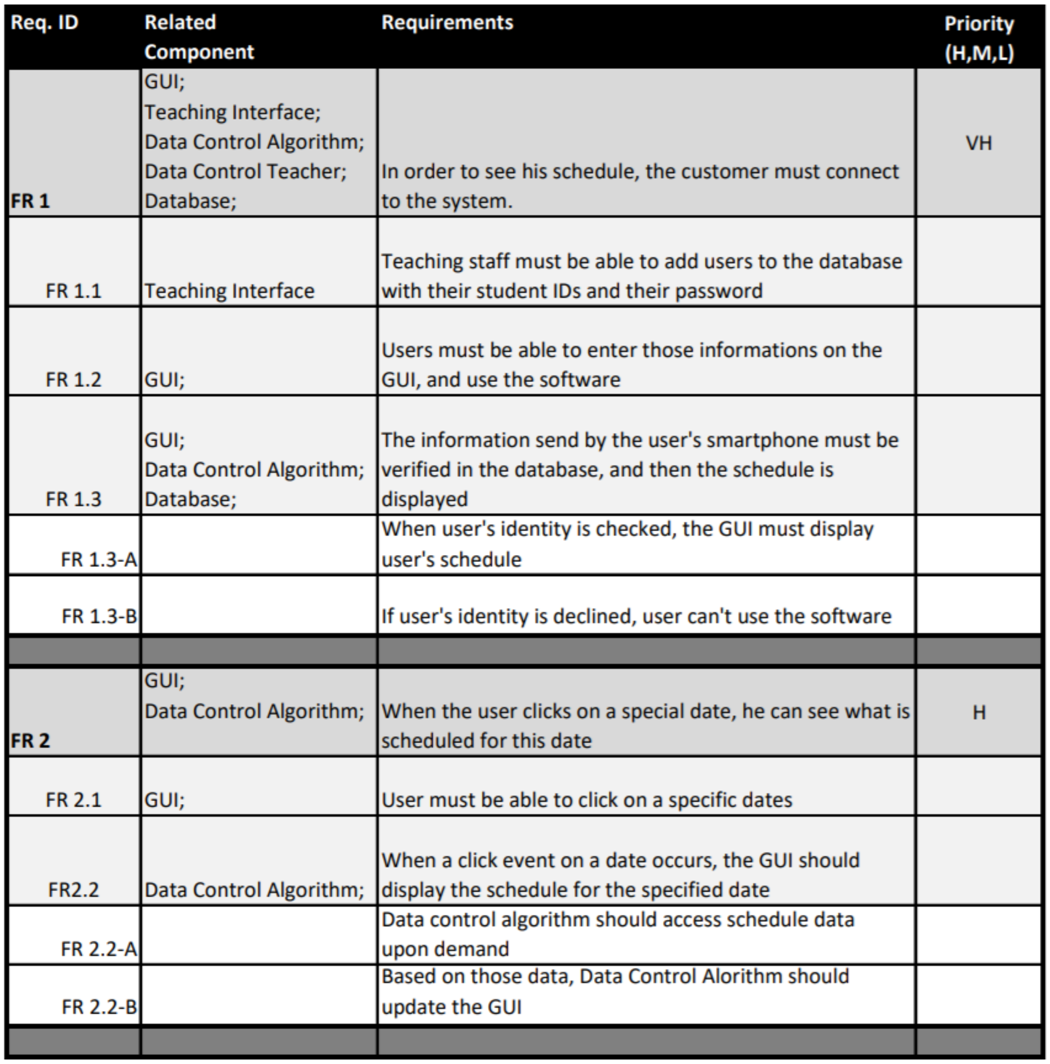
- The system depends on the database
- The system depends on teaching staff updating the class schedule when there's change.
- The system depends on students to update information about an upcoming event if there is a change
- The system will use a calendar-typed API, hence it will depend on the availability of the service

3 System Features

In this section we will present the Functional requirements. Based on that we created a Use case Diagram and a sequence diagram for the three main use cases and a domain model diagram.

3.1 Functional Requirements

We summarized the different Functionnal Requirements into a requirements table:



Req. ID	Related Component	Requirements	Priority (H,M,L)

FR 1	GUI;
	Teaching Interface;
	Data Control Algorithm;
	Data Control Teacher;
	Database;
		In order to see his schedule, the customer must connect to the system.	VH

FR 1.1	Teaching Interface	Teaching staff must be able to add users to the database with their student IDs and their password	
FR 1.2	GUI;	Users must be able to enter those informations on the GUI, and use the software	

FR 1.3	GUI;
	Data Control Algorithm;
	Database;
		The information send by the user's smartphone must be verified in the database, and then the schedule is displayed	

FR 1.3-A		When user's identity is checked, the GUI must display user's schedule	
FR 1.3-B		If user's identity is declined, user can't use the software	

FR 2	GUI;
	Data Control Algorithm;
		When the user clicks on a special date, he can see what is scheduled for this date	H

FR 2.1	GUI;	User must be able to click on a specific dates	
FR2.2	Data Control Algorithm;	When a click event on a date occurs, the GUI should display the schedule for the specified date	
FR 2.2-A		Data control algorithm should access schedule data upon demand	
FR 2.2-B		Based on those data, Data Control Alorithm should update the GUI	

FR 3	GUI;
	Data Control Algorithm;
	Database;
		User should be able to create and share an event	~H

FR 3.1	GUI;
	Data Control Algorithm;
	Database;
		User should be able to create an event on the GUI	H

FR 3.1-A	GUI;	User should be able to enter precise information on the event : Hours, type, name, place, (private or public)	

FR 3.1-B	Data Control Algorithm;
	Database;
		Data Control should store this informations locally and send them to the Database	

FR 3.2	GUI;
	Data Control Algorithm;
	Database;
		User should be able to share an event on the GUI	M

FR 3.2-A	GUI;	User can tag people on one event	

FR 3.2-B	Data Control Algorithm;
	Database;
		Data Control should update the Participant of one event in the Database	

FR 4	GUI;
	Data Control Algorithm;
	Database;
		User should be able to search open event and modify his schedule if he accepts one event he has been invited to.	L

FR 4.1	GUI;	User should be able to search an event using the GUI	

FR 4.1-A	Data Control Algorithm;
	Database;
		Data Control Algorithm should retrieve specific event from database	

FR 4.1-B	Data Control Algorithm;
	GUI;
		Data Control Algorithm should update GUI	

FR 4.2	GUI;	User should be able to select and join an event	

FR 4.2-A	Data Control Algorithm;
	Database;
		Data Control Algorithm should send the selected event to the database	

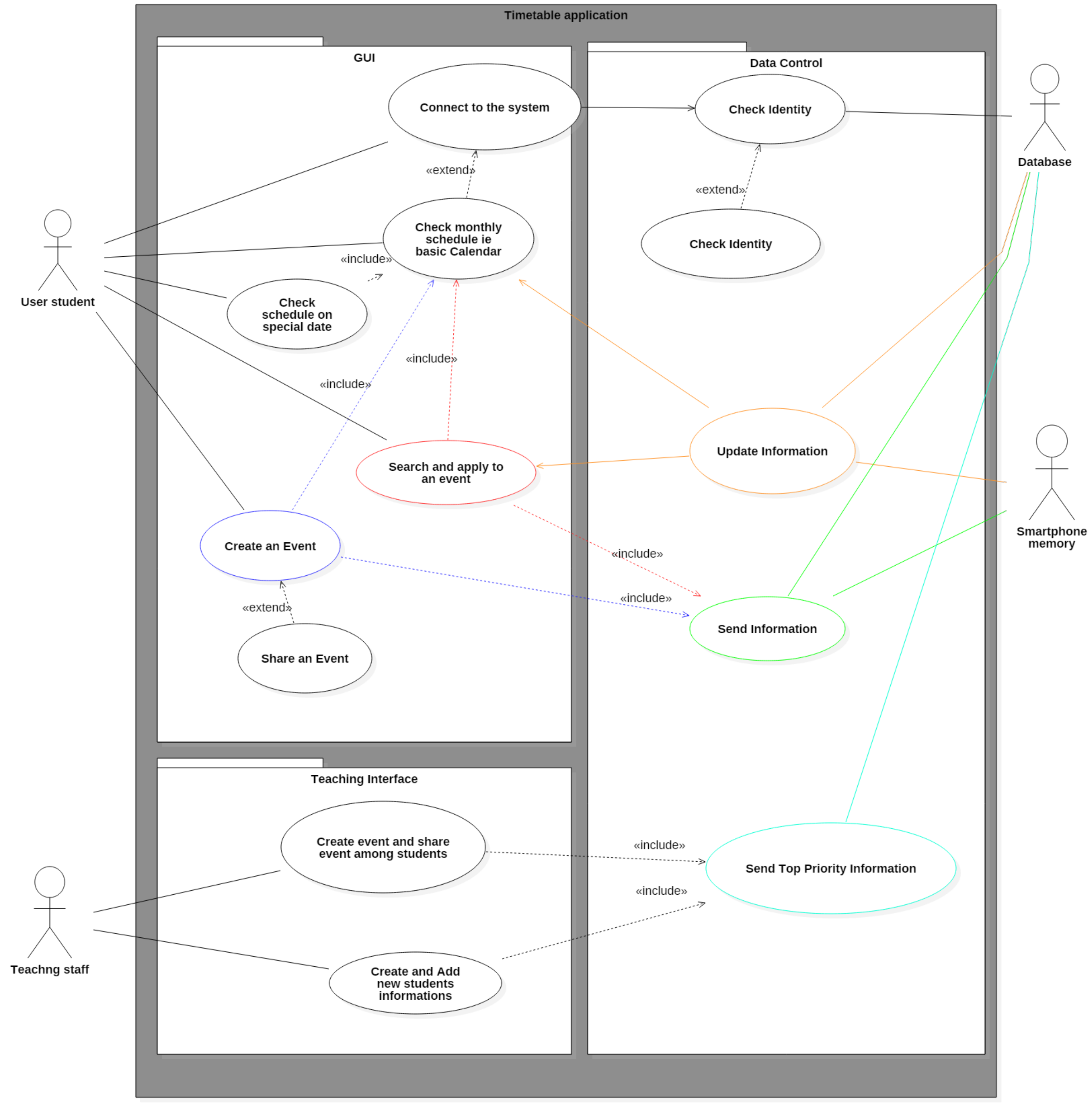
FR 4.2-B	Data Control Algorithm;
	GUI;
		Data Control Algorithm should update the schedule on the GUI	

FR 5	Teaching Interface;
	Data Control Teacher;
	Database;
		Teacher should be able to add an event (Assignment, extra class hours), and include all participant	H

	Same requirements for	FR 5 and FR 3.	
FR 6	Database; Data control Algorithm	Schedule data should be loaded and updated on the GUI asynchronously.	L
FR 6.1	Database; Data control Algorithm	Data Control Algorithm should fetch on a given time data from the Database	
FR 6.2	Database; Data control Algorithm	Data Control Algorithm should update the GUI on a given time	
FR 7	Data Control Algorithm;	If there's an upcoming event, the User should be notified	L
FR 7.1		Data Control Algotihm detects if there's an upcoming event	
FR 7.2		Data Control Algorithm push a notification	

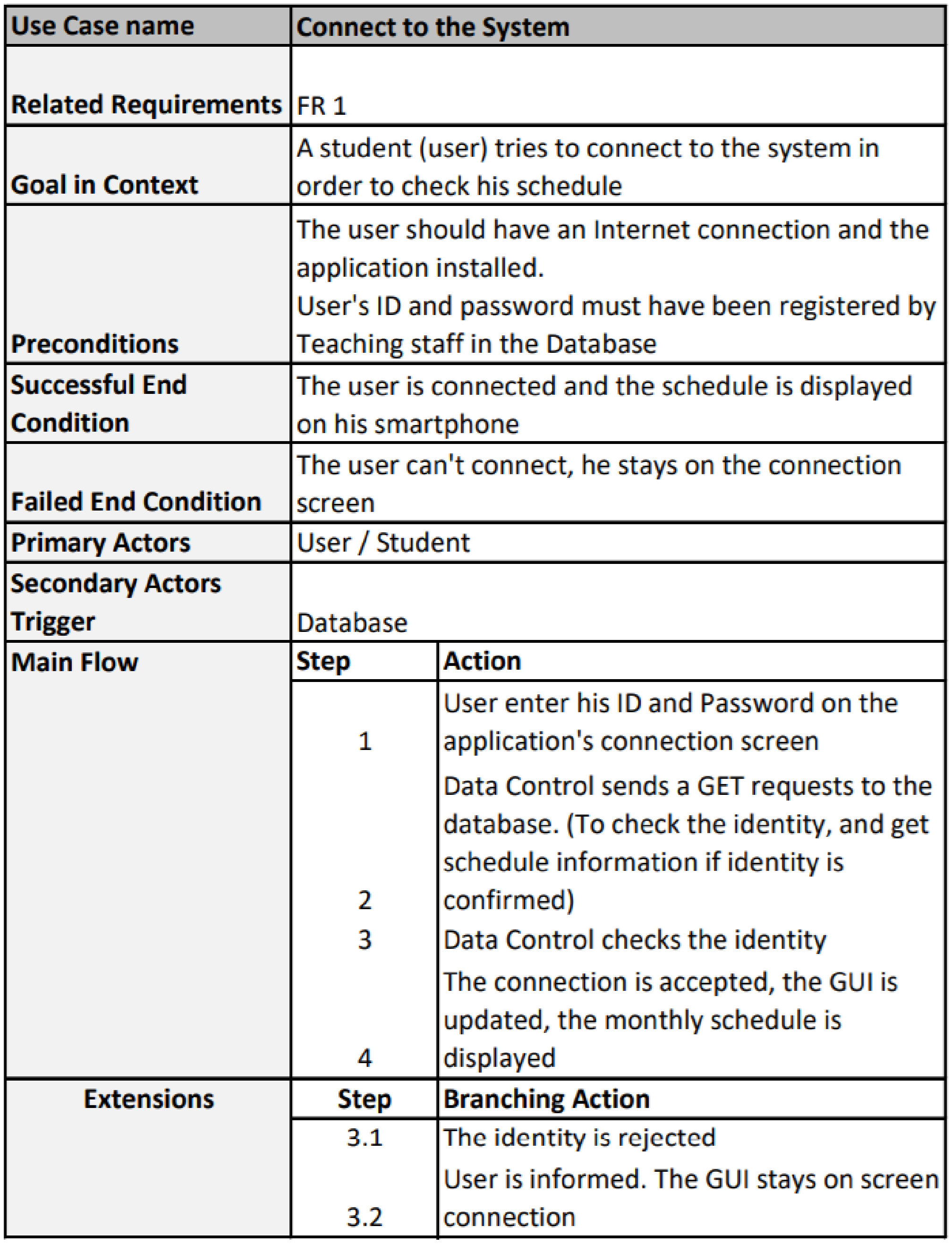
3.2 Use Case Diagram & Requirements

.



The three main use cases are the one related to FR1, FR2 and FR 3. In the following table, we use the term ”monthly schedule”. This refers to the main Activity (ie the main Screen) the user will deal with by using the application.

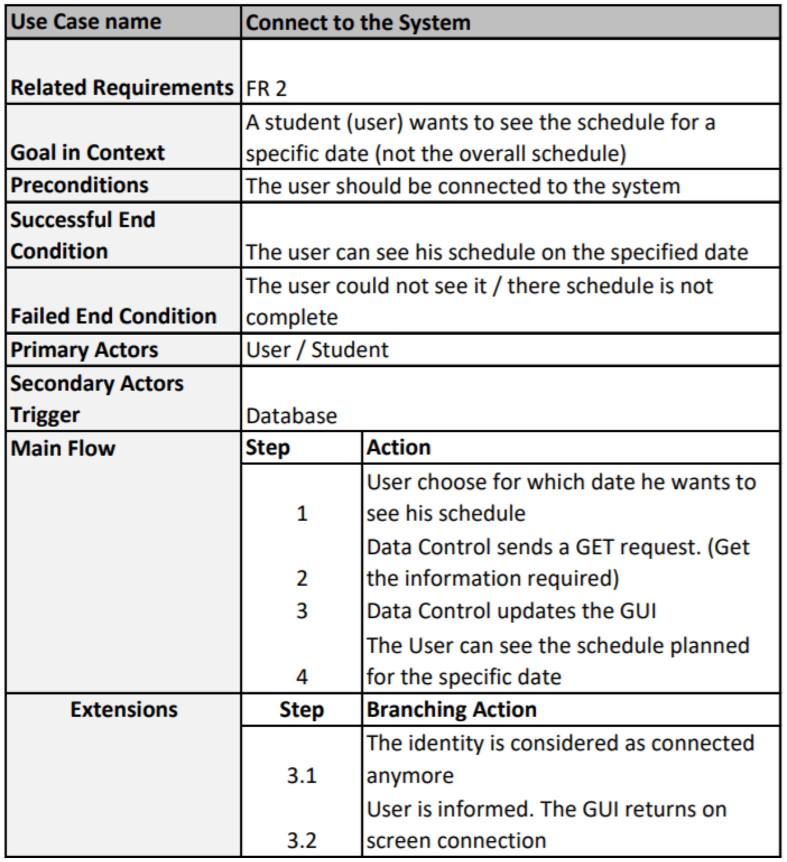
Use case FR1:



Use Case name	**Connect to the System**	
Related Requirements	FR 1	
Goal in Context	A student (user) tries to connect to the system in order to check his schedule	
Preconditions	The user should have an Internet connection and the application installed. User's ID and password must have been registered by Teaching staff in the Database	
Successful End Condition	The user is connected and the schedule is displayed on his smartphone	
Failed End Condition	The user can't connect, he stays on the connection screen	
Primary Actors	User / Student	
Secondary Actors Trigger	Database	
Main Flow	**Step**	**Action**
	1	User enter his ID and Password on the application's connection screen
	2	Data Control sends a GET requests to the database. (To check the identity, and get schedule information if identity is confirmed)
	3	Data Control checks the identity
	4	The connection is accepted, the GUI is updated, the monthly schedule is displayed
Extensions	**Step**	**Branching Action**
	3.1	The identity is rejected
	3.2	User is informed. The GUI stays on screen connection

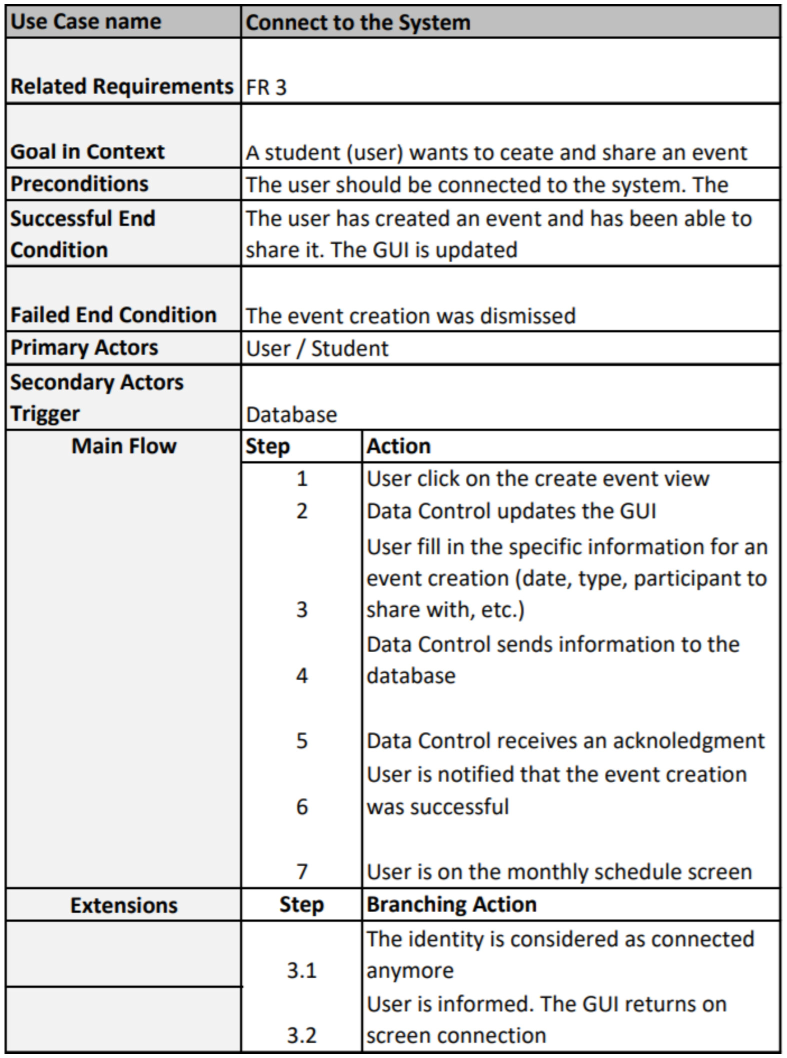
Figure 3: Use Case FR1

Use case FR2 and FR3:



Use Case name	Connect to the System	
Related Requirements	FR 2	
Goal in Context	A student (user) wants to see the schedule for a specific date (not the overall schedule)	
Preconditions	The user should be connected to the system	
Successful End Condition	The user can see his schedule on the specified date	
Failed End Condition	The user could not see it / there schedule is not complete	
Primary Actors	User / Student	
Secondary Actors Trigger	Database	
Main Flow	Step	Action
	1	User choose for which date he wants to see his schedule
	2	Data Control sends a GET request. (Get the information required)
	3	Data Control updates the GUI
	4	The User can see the schedule planned for the specific date
Extensions	Step	Branching Action
	3.1	The identity is considered as connected anymore
	3.2	User is informed. The GUI returns on screen connection

(a) Use Case FR 2



Use Case name	Connect to the System	
Related Requirements	FR 3	
Goal in Context	A student (user) wants to ceate and share an event	
Preconditions	The user should be connected to the system. The	
Successful End Condition	The user has created an event and has been able to share it. The GUI is updated	
Failed End Condition	The event creation was dismissed	
Primary Actors	User / Student	
Secondary Actors Trigger	Database	
Main Flow	Step	Action
	1	User click on the create event view
	2	Data Control updates the GUI
	3	User fill in the specific information for an event creation (date, type, participant to share with, etc.)
	4	Data Control sends information to the database
	5	Data Control receives an acknoledgment
	6	User is notified that the event creation was successful
	7	User is on the monthly schedule screen
Extensions	Step	Branching Action
	3.1	The identity is considered as connected anymore
	3.2	User is informed. The GUI returns on screen connection

(b) Use Case FR3

Figure 4

3.3 Domain Model

The figure bellow (**Figure 5**) describe our domain model

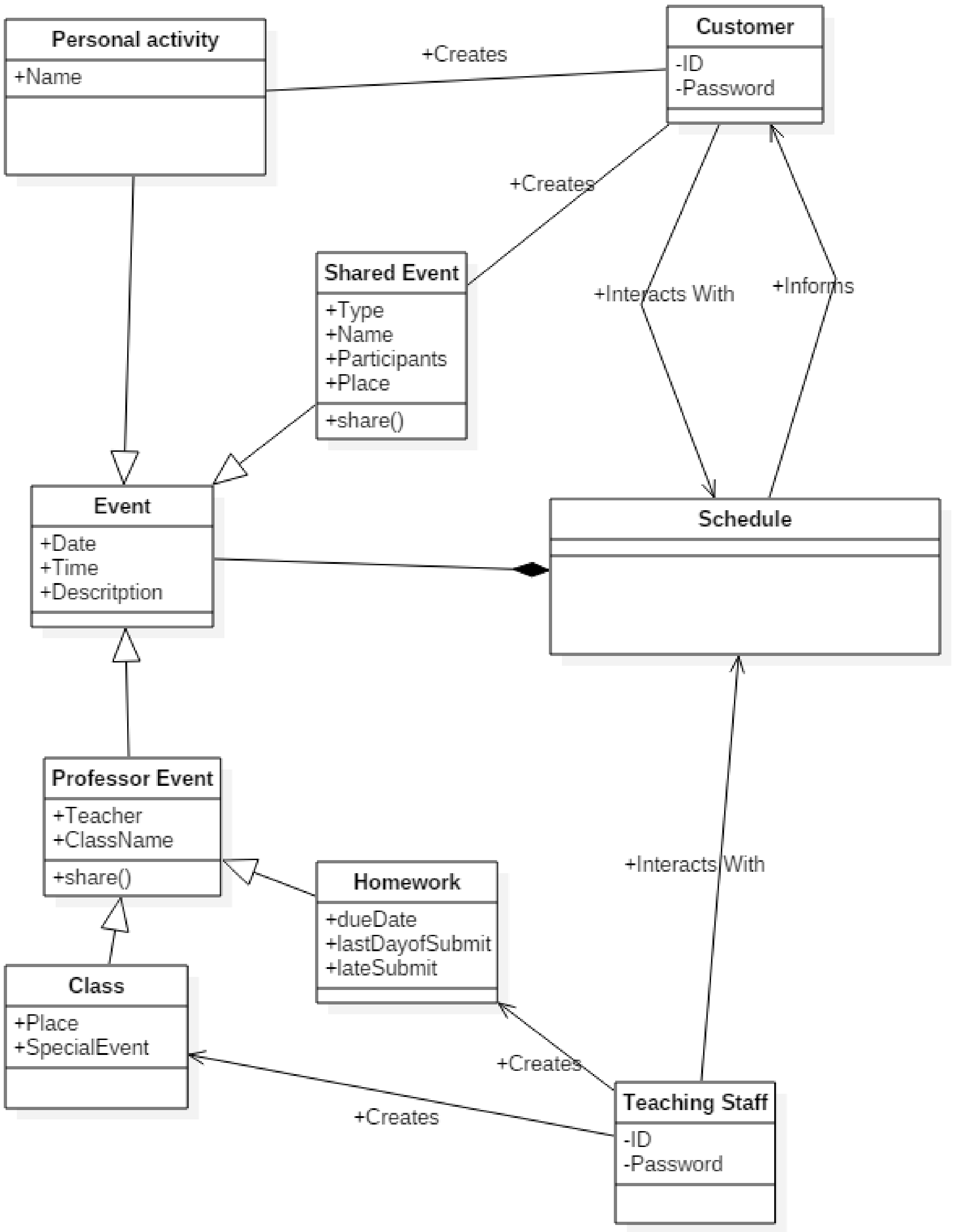


Figure 5: Domain model

3.4 Sequence Diagrams

You'll find bellow the sequence diagram of teh three main use case

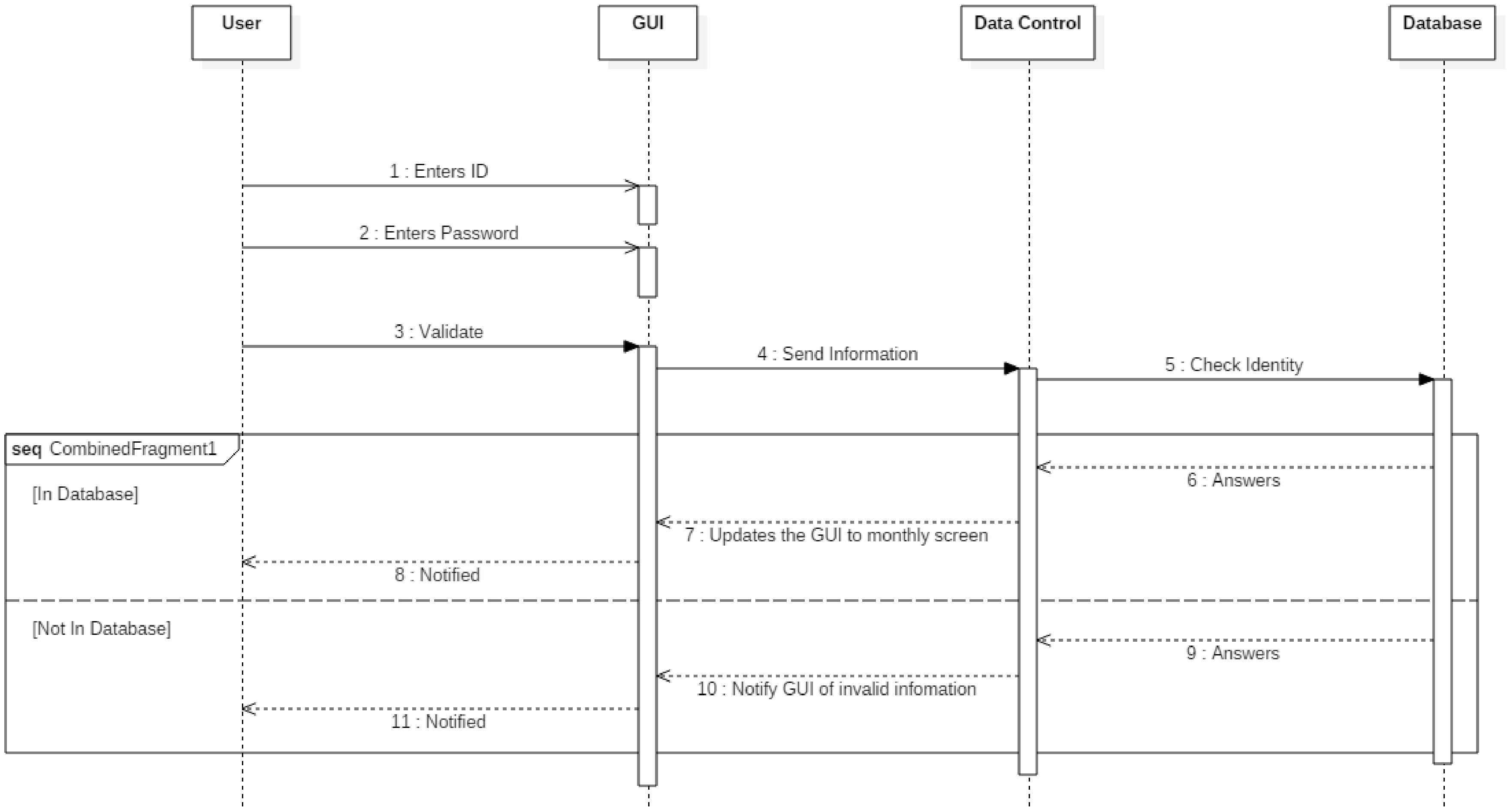


Figure 6: Sequence Diagram : User connects to the system

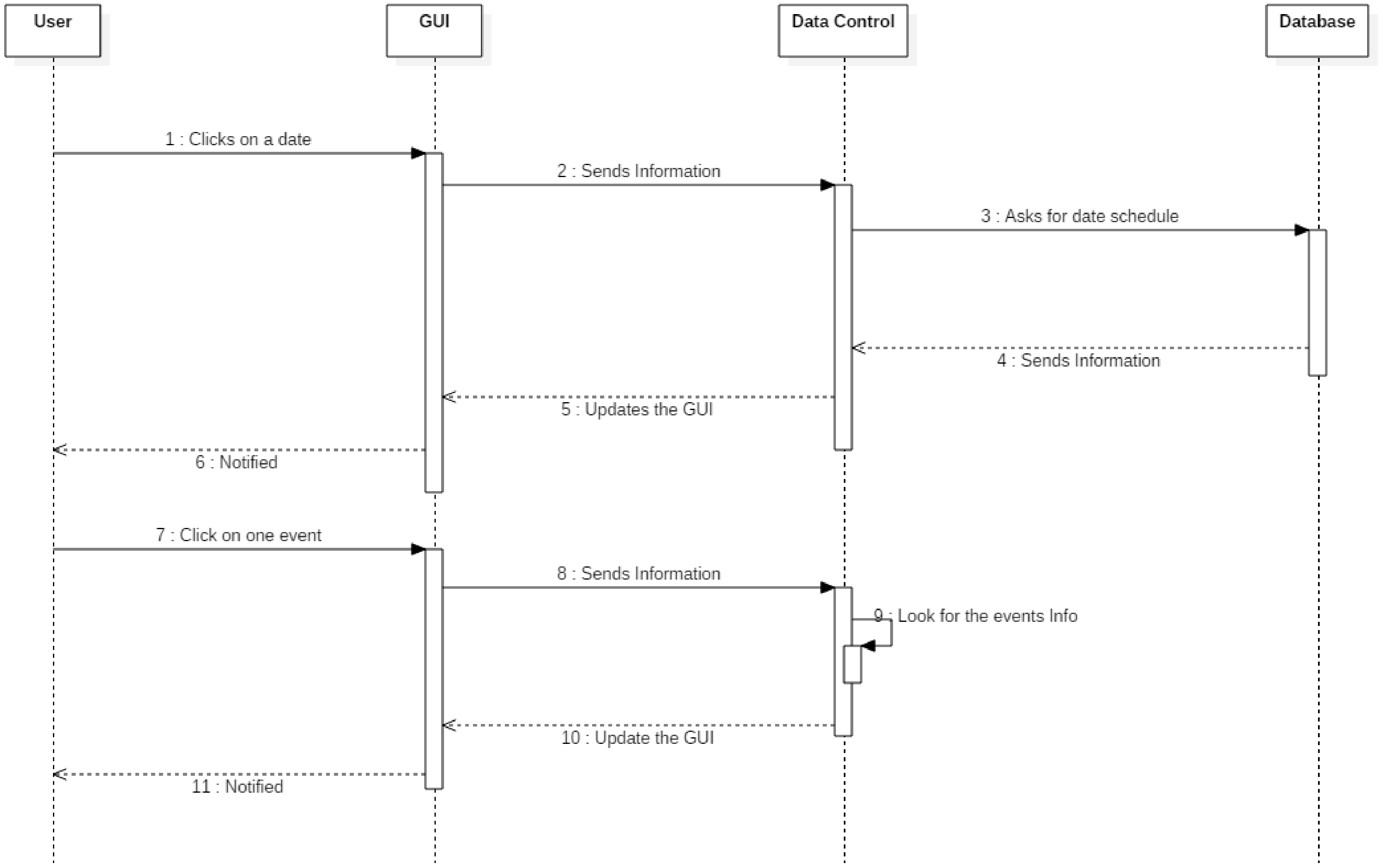


Figure 7: Sequence Diagram : User wants to see his schedule on a specific date

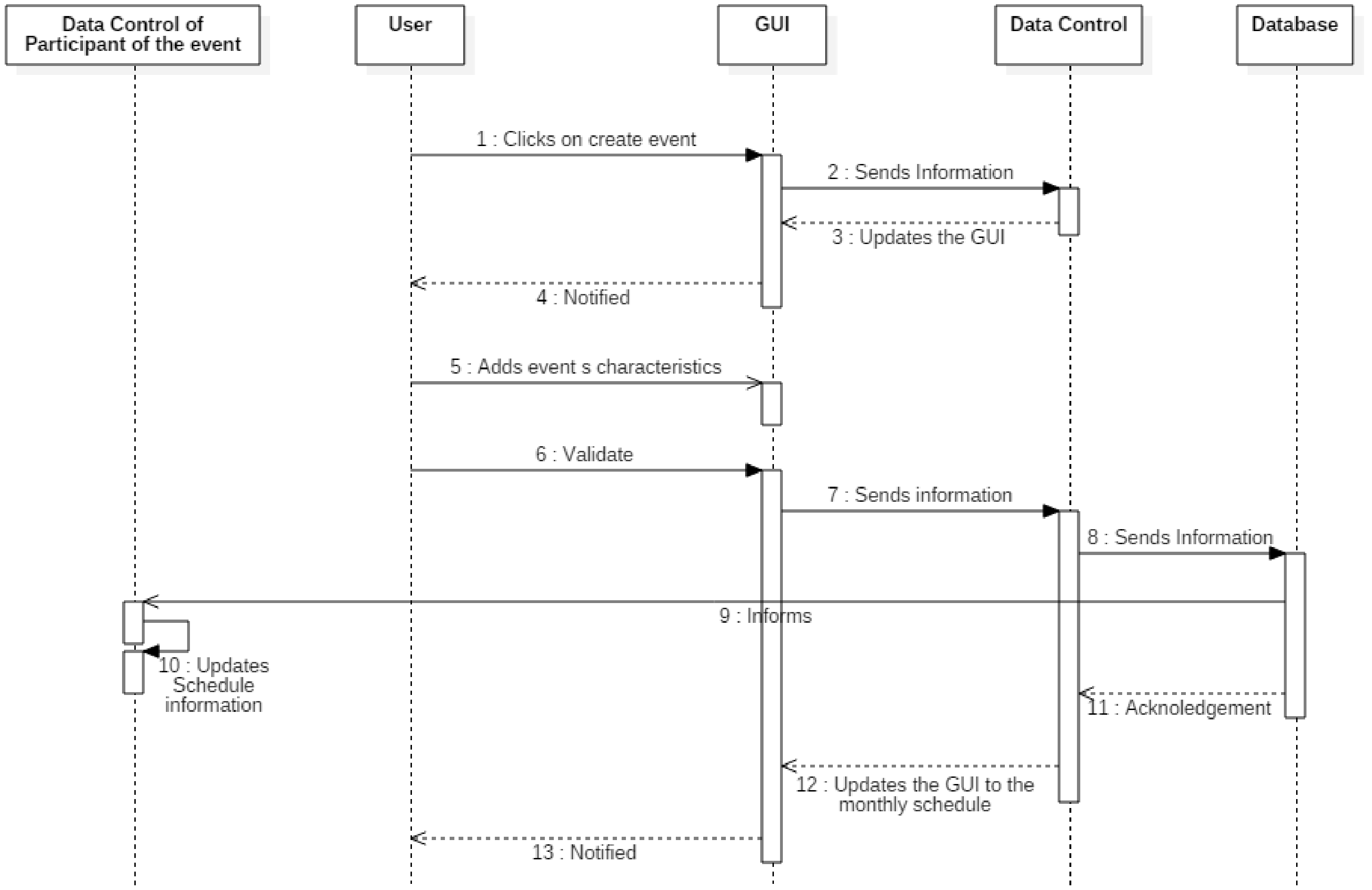


Figure 8: Sequence Diagram : User creates and share an event

4 Preliminary User Manual

The user of this part of the manual is the same person who uses the timetable application. We will focus on the main function of our system. This means : see the monthly schedule, see the schdule on a specific date and make an event. We provide preliminary screenshots made on PowerPoint to give you an idea of what it will look like. However the design of the interface might change a lot during the developpment.

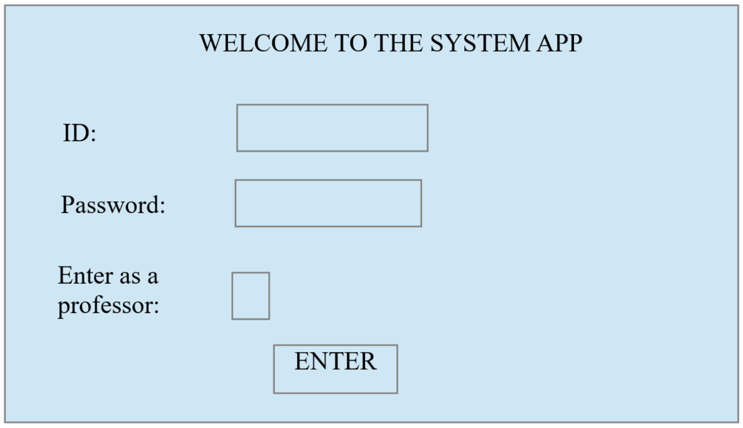
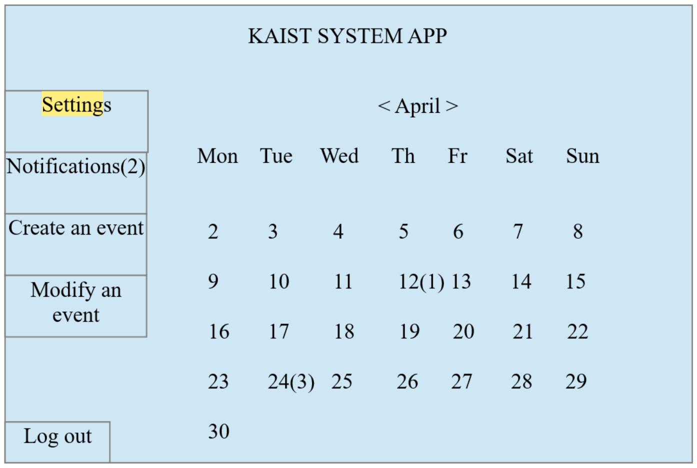


Figure 9: Connection Screen

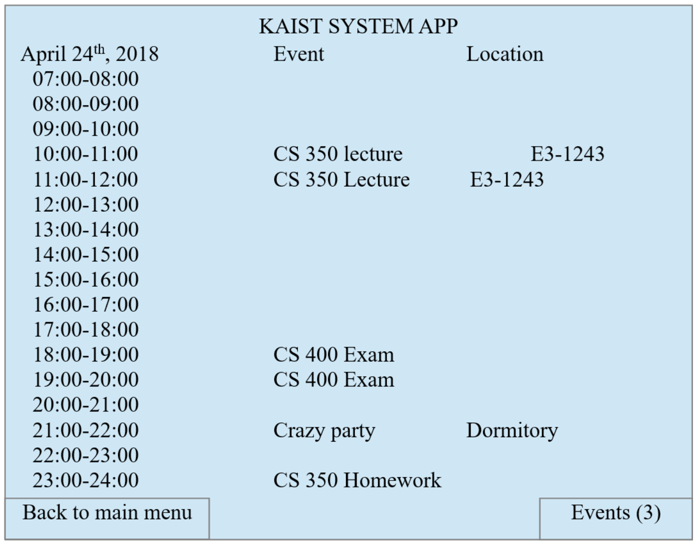
User of the manual will be the KAIST students and KAIST professors. KAIST students read the manual to use the app for scheduling their duties, while professors will use it to add their courses' tasks. Initial connection interface will be similar to **Figure 9**.If the correct ID and password pair is entered the user can enter the system. Professors should enter the system by filling the checkbox denoted as "Enter as a professor". Then they will have some extra options compared to the normal users (students).

After a correct entrance, users will meet with the interface on **Figure 10**.Notifications screen is similar to the alarm bell on the KLMS. It will denote the number of new events, events that are supposed to take place today. In addition, it will contain a warning message if there are multiple events at the same day, same hour in the future dates including today. Both functionalities, create an event and log out buttons, talk by themselves. User can modify an event by clicking the respective button only if they have the permission to modify that event.



Mon	Tue	Wed	Th	Fr	Sat	Sun
2	3	4	5	6	7	8
9	10	11	12(1)	13	14	15
16	17	18	19	20	21	22
23	24(3)	25	26	27	28	29
30						

Figure 10: Main Screen



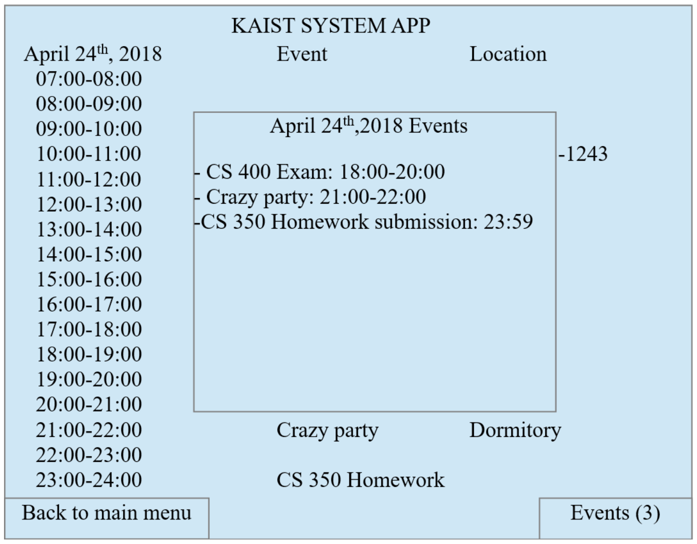
KAIST SYSTEM APP

April 24th, 2018	Event	Location
07:00-08:00		
08:00-09:00		
09:00-10:00		
10:00-11:00	CS 350 lecture	E3-1243
11:00-12:00	CS 350 Lecture	E3-1243
12:00-13:00		
13:00-14:00		
14:00-15:00		
15:00-16:00		
16:00-17:00		
17:00-18:00		
18:00-19:00	CS 400 Exam	
19:00-20:00	CS 400 Exam	
20:00-21:00		
21:00-22:00	Crazy party	Dormitory
22:00-23:00		
23:00-24:00	CS 350 Homework	

Figure 11: Event Screen

On the calendar, days with some events will have number of events within parentheses next to them, like 12 and 24 of April. To see the events, user must click on the day and a new menu for the will appear. You can see below for an example. To change month, user must press on the "<" or ">" signals next to the name of the month. **Figure 11** shows how the event screen will look like.

User can see the events of that day listed by their starting time on a pop-up window by clicking on the events button. **Figure 12** shows how it will look like.

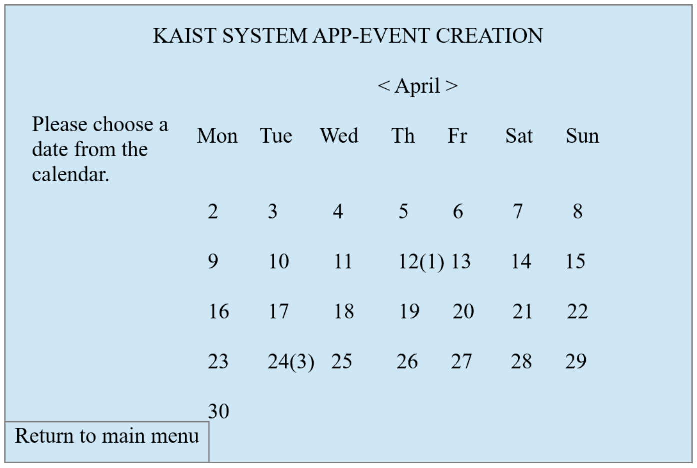


KAIST SYSTEM APP

April 24th,2018 Events

- CS 400 Exam: 18:00-20:00
- Crazy party: 21:00-22:00
-CS 350 Homework submission: 23:59

Figure 12: Event Screen



KAIST SYSTEM APP-EVENT CREATION

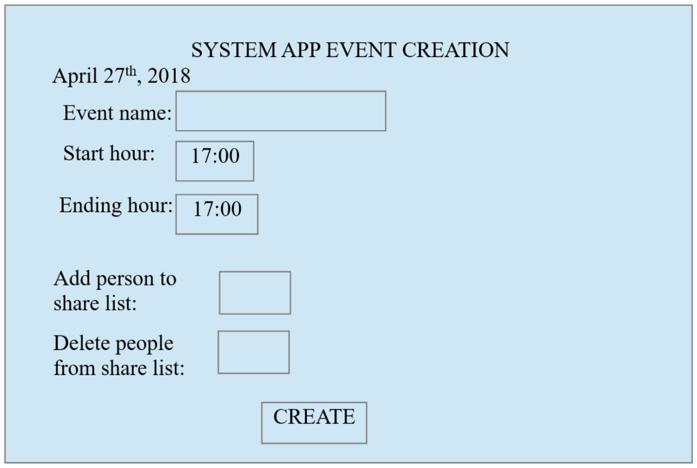
< April >

Please choose a date from the calendar.

Mon	Tue	Wed	Th	Fr	Sat	Sun
2	3	4	5	6	7	8
9	10	11	12(1)	13	14	15
16	17	18	19	20	21	22
23	24(3)	25	26	27	28	29
30						

Return to main menu

Figure 13: Event Creation First Screen



SYSTEM APP EVENT CREATION

April 27th, 2018

Event name:

Start hour: 17:00

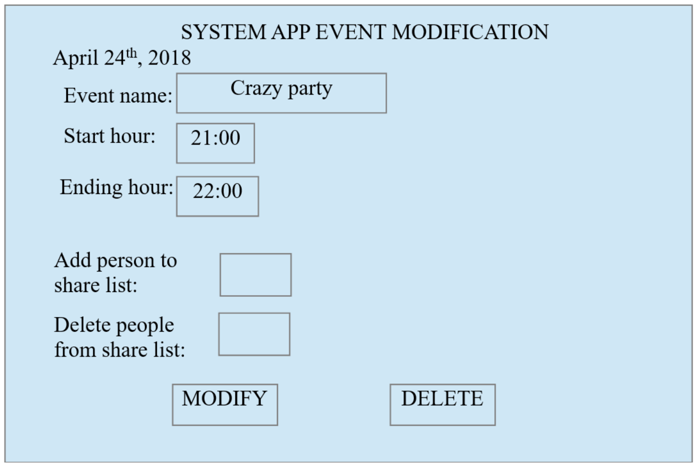
Ending hour: 17:00

Add person to share list:

Delete people from share list:

CREATE

Figure 14: Event Creation Second Screen



SYSTEM APP EVENT MODIFICATION

April 24th, 2018

Event name: Crazy party

Start hour: 21:00

Ending hour: 22:00

Add person to share list:

Delete people from share list:

MODIFY DELETE

Figure 15: Event Creation Third Screen

The user who wants to create an event clicks on the date they want to create the event (**Figure 13**). A new menu appears with specifications for the event. Suppose the user chooses 27th of April. User can choose the name, hours and the people they want to share the event with (**Figure 14**). By creating an event, people participating the event will have their schedule updated automatically. If user does not want to share the event with an other user , he just chooses a person from the drop down menu and when the person is chosen, he/she is automatically deleted from the list. Professors can add whole classes, like CS 350 class to the share list. So when the event is created, it is shared with all the students taking that course. When a new event is shared, it is directly added to the schedules. Also, a pop-up window appears informing users about the event. Moreover, the pop-up window let the user decide whether he wants or doesn't want to join the shared event. This choice occurs only if event is created by a student, not by a professor. The process to modify an event is similar to creating an event. The user will choose the date of the event he wants to modify. Then he will choose the event. And finally he will face the interface shown in (**Figure 14**). From this interface, user can change the name, hours of the event or can even delete it. But to avoid abuses, only the creator of an event can modify or delete an event.

5 Acceptance Criteria

The checklist bellow shows what tests the finished product should pass. The letters between brackets (VH,H,M,L,VL) correspond to the priority of the test. Before the end of the semester, our software should pass at least the (VH) and some of the (H).

☐ **(VH)** The User should be able to enter the application

- Test input : Access to the applicaiton
- Expected Output : Connection screen displayed

☐ **(VH)** The User should be able to connect to the system if he has an account registered

- Test input : Valid combination ID, password
- Expected output : Main Screen (Monthly Schedule) Displayed

☐ **(VH)** The User should not be able to connect if not registered in the database

- Test input : Wrong ID or Wrong password for a valid ID
- Expected output : Notify the user that the connection has failed due to authentification issues. Stays on conneciton screen

☐ **(H)** The User should be able to check one event of a specific date

- Test input : The user click on one date of the calendar
- Expected output : Schedule of that date, with all the information possible

☐ **(H)** The User should be able to add an event.

- Test input : Test input : User enters information of one event (Name, Date, Type, etc.)
- Expected output : Event successfully added in the database

☐ **(M)** The User should be able to add an event and share it

- Test input : User enters information of one event (Name, Date, Type etc.) and Participants' ID
- Expected output : Event succssfully added in the database and shared with the participants

☐ **L** The User should be able to search for an open event

- Test input : User searches for one event
- Expected output : Search screen display the open events

☐ **VL** The User should be able to search for an open event and add it to its schedule

- Test input : User searches for one event and adds it to its schedule
- Expected output : Event added to user's schedule in the database

☐ **VL** The User should be notified of an upcoming event

- Test input : Upcoming event
- Expected output : Push notification

6 Non-functional Requirments (Quality Attribute)

A. Accessibility

- [NF.ACC-1, **M**] The application should be functional even when internet connection is offline. However, connection is required to fetch shared data from server.

B. Availability

- [NF.AVA-1, **L**] The application should be available 24/7.

C. Backup

- [NF.BCK-1, **H**] All data are stored in individual smartphone.
- [NF.BCK-2, **H**] Shared data will also be stored in distant database. If the information of shared data is different at individual smartphone database, the application will modify the individual database.

D. Data retention and privacy

- [NF.DRP-1, **H**] Individual data will only be used for this application.
- [NF.DRP-2, **H**] Individual schedules cannot be seen or changed by other users unless the creator of the schedule marks it as shared.
- [NF.DRP-3, **H**] Schedules for homework and projects can be seen by class members if created by faculty members. However, it cannot be modified by class members.

E. Safety and fault tolerance

- [NF.SFT-1, **L**] Systems will send notifications before a schedule, with respect to the time and date of user's phone, which can differ from standard time.
- [NF.SFT-2, **M**] Any mistakes while creating, modifying a schedule can be modified.

F. Platform and deployment environment

- [NF.PDE-1, **L**] The application will run on android devices and IOS devices.
- [NF.PDE-2, **L**] The application will be deployed through Playstore and Appstore.

G. Maintainability and quality

- [NF.MAQ-1, **H**] Any errors or faults in the system should be noticed immediately and repaired within a day.

H. Performance characteristics

- [NF.PER-1, **M**] Response time of all functionalities should be less than 2 seconds.
- [NF.PER-2, **M**] The application should not affect device performance significantly.

I. Usability and readability

- [NF.USR-1, **M**] The capability of the application should clear and well understood from the UI.

J. Interoperability

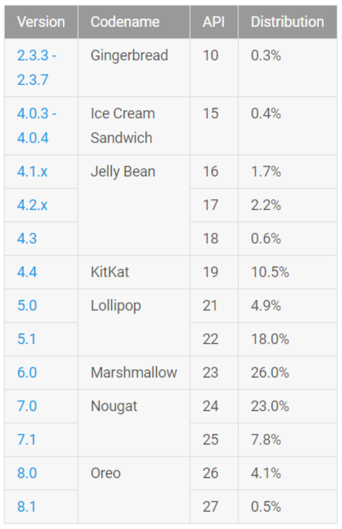
- [NF.INT-1, **M**] The application should be able to run on background in order to synchronize shared data.

K. Open Source

- [NF.OPS-1, **L**] Source code should be available to all.

7 Appendix

Android usage: Screen size:



Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.7%
4.2.x		17	2.2%
4.3		18	0.6%
4.4	KitKat	19	10.5%
5.0	Lollipop	21	4.9%
5.1		22	18.0%
6.0	Marshmallow	23	26.0%
7.0	Nougat	24	23.0%
7.1		25	7.8%
8.0	Oreo	26	4.1%
8.1		27	0.5%

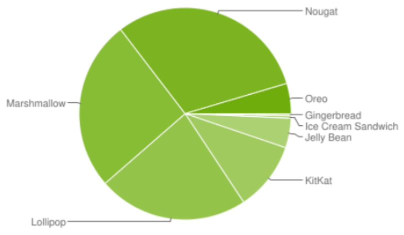
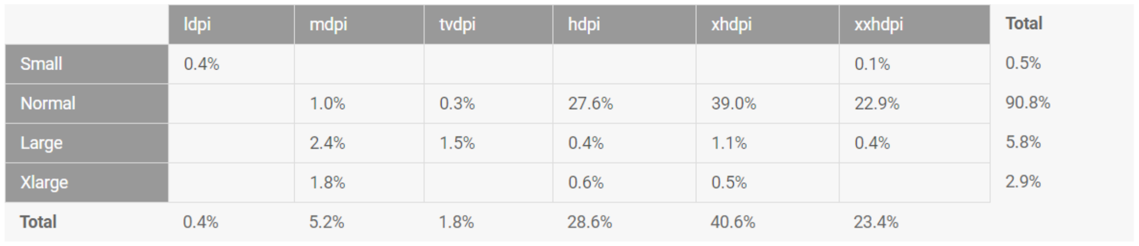


Figure 16: Distribution of Android's usage with respect to the Operating System
Sources : https://developer.android.com/about/dashboards/index.html



	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	0.4%					0.1%	0.5%
Normal		1.0%	0.3%	27.6%	39.0%	22.9%	90.8%
Large		2.4%	1.5%	0.4%	1.1%	0.4%	5.8%
Xlarge		1.8%		0.6%	0.5%		2.9%
Total	0.4%	5.2%	1.8%	28.6%	40.6%	23.4%	

Figure 17: Distribution of Screen sizes among Android Smartphones
Sources : https://developer.android.com/about/dashboards/index.html

8 Acknoledgment

Editor of this document : Alexandre Allani

Redactors:

- Introduction : Alexandre Allani
- Overall Perspective : Alexandre Allani
- System Features : Alexandre Allani
- Preliminary User Manual : Bilgehan Bingol
- Acceptance Criteria : Alexandre Allani
- Non-functional Requirements (Quality Attribute) : Shin Seung Hun and Kuntae Park
- Appendix : Alexandre Allani