Team Project SRS Team #8

Spring 2018

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

June 7, 2018

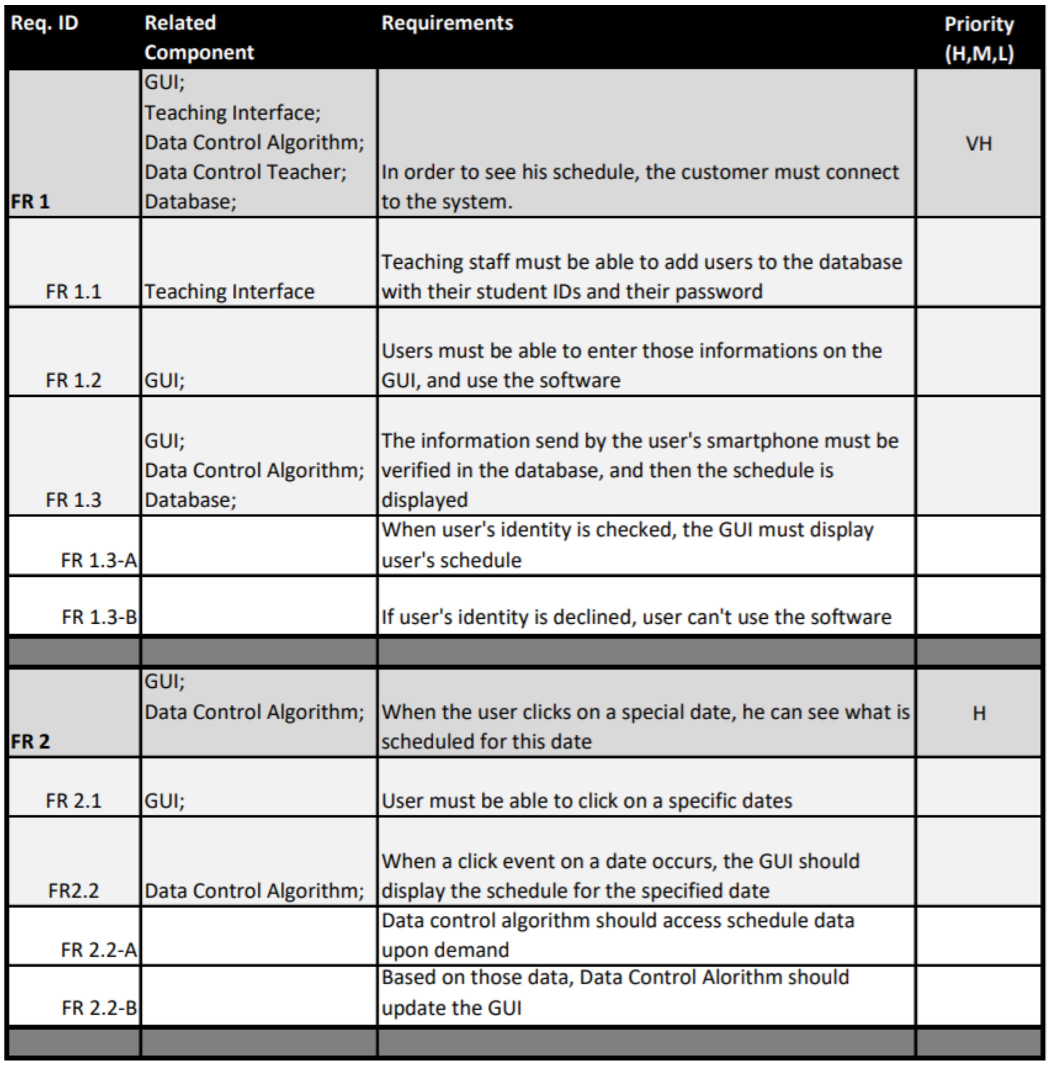
This document is the Software Requirements Specification of our project.

Contents

1 System Overview

1.1 Requirements

We summarized the different Functionnal Requirements into a requirements table (copy of SRS document):



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	

1.2 Tasks

Tasks FR5 not included since it rest upon the same base as FR3.

- T.1: Implement graphical user interface/ FR1, FR2, FR3, FR4, NF.PDE-1, NF.USR-1
  - T.1:.1. Implement user authentication UI/ FR1.1, FR1.2
  - T.1:.2. Implement schedule timetable/calendar UI/ FR1.3, FR2.1,
  - T.1:.3. Implement schedule editing/ UI FR3.1, FR3.2
  - T.1:.4. Implement sharable event searching UI/ FR4.1, FR4.2
- T.2: Implement control system/ FR2, FR3, FR4, FR7, NF.ACC-1, NF.BCK-1, NF.PDE-1, NF.PER-1, NF.PER-2, NF.INT-1
  - T.2:.1. Implement individual schedule editing system/ FR2.2, FR3.1, FR3.2, NF.DRP-2, NF.DRP-3, NF.SFT-2
  - T.2:.2. Implement event notification system/ FR 7, NF.SFT-1
  - T.2:.3. Implement sharable event searching algorithm/ FR4.1, 4.2
- T.3: Test control system
- T.4: Connect control system with the GUI/ FR6
- T.5: Ready database
- T.6: Implement database connections/ FR1, FR3, FR4, FR6, NF.BCK-2, NF.PER-1, NF.INT-1
  - T.6:.1. Implement user authentications with database information/ FR1.3
  - T.6:.2. Implement shared schedule data fetching/ FR3.2, FR4.1, FR4.2
- T.7: Test whole system

2 System Design

In this section we will take look at our system and how we imagine it to work.

2.1 System Architecture

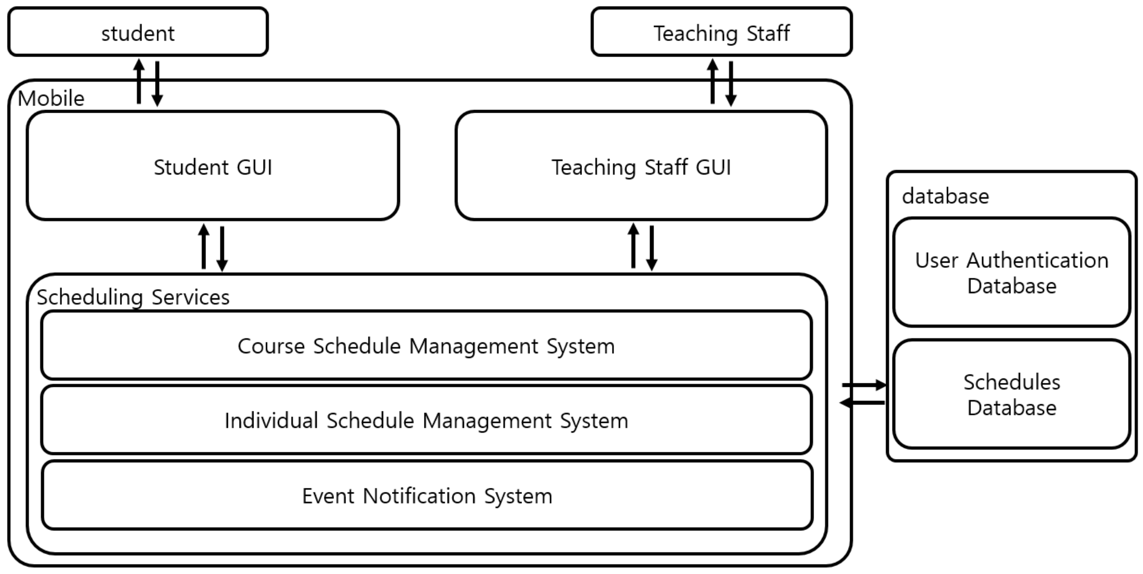

student
Teaching Staff
Mobile
Student GUI
Teaching Staff GUI
database
User Authentication Database
Scheduling Services
Course Schedule Management System
Individual Schedule Management System
Event Notification System
Schedules Database

Figure 1: Overview of our application

As you can see on **Figure 1**, we will use this architecture to developp the application. Both student and Teacher assistant will use their own GUI. Their interaction will be transmitted to the Scheduling Services
Scheduling is divided in 3 parts:

- *Course Schedule Management System:* This part will process the different classes that the user is taking.
- *Individual Schedule Management System:* This part will process the personnal schedule and what is related to it.
- *Event Notification System:* This part will alert the user when there's an upcoming event.

2.2 Class Diagram

2.2.1 List of classes

All the described classes are on diagrams, on next pages.
Model:

- **Event** : This class is an abstract class, meaning it can't be instanciated. It contain the basic information, comon among all the event-typed class. The boolean *priority* is here to know if the user is a student or a teacher. In case of the teacher, the event might override some of the student events. The function $deletePerson()$ remove a user from the event. $addPerson()$ do the opposite.
- **TeacherEvent**: This is also an abstract class. It inherits from **Event**.It contains the basic information of a teacher typed event.
- **Course** : This class can be instanciated. It inherits from **TeacherEvent** . It contains all the information needed for a course typed event.
- **HomeworkEvent** : This class can be instanciated. It inherits from **TeacherEvent**.It contains all the information needed for a homework typed event.
- **Person** : This class can be instanciated and contains all the informations of the user. It might be also instanciated for other people.
- **Schedule**: This class is the one that links everything. $addEvent()$ and $deleteEvent()$ respectively add and delete a special event on user's schedule. $shareEvent()$ and $modifyEvent()$ respectively share and modify and event which already exists.

Refer to the next subsection to have a better look at views.
View:

- **ConnectionScreen**: Contains all the view of the connection Screen
- **MainScreen**: Contains all the view of the main Screen
- **EventScreen**: Contains all the view of the event Screen
- **EventFirstCreationScreen**: Contains all the view of the first screen of an event's creation
- **EventSecondCreationScreen**:Contains all the view of the second screen of an event's creation
- **EventModifyScreen**: Contains all the view of the modify event screen. It inherits from the $EventSecondCreationScreen$. Indeed, the screen is pretty much the same, just two buttons are different.
- **TeacherEventScreen**: Contains all the view of the first event screen of a teacher event's creation
- **CourseScreen**: Contains all the view of Course's screen creation

- **EventModifyCourseScreen**: Contains all the view of the modify event screen of a course creation. It inherits from the *CourseScreen*. Like the other modify event screen, just two buttons change.
- **HomeworkScreen**: Contains all the view of homework's screen creation
- **EventModifyHomeworkScreen**: Contains all the view of the modify event screen of a homework creation. It inherits from the *HomeworkScreen*. Like the other modify event screen, just two buttons change.

All of the Controller's classes are the counterpart of each View's classes. Basically, they just modify the Screen and data of the model according to what the user do with the GUI. All the class inherit from the class Controller, which has as an attribute the model.
Controller:

- **ConnectionController**: According to the button pressed, change the screen to *MainScreen* or *ConnectionScreen*
- **MainScreenController**: According to the button pressed, change the screen to *EventScreen* or *TeacherEventScreen* or *EventFirstCreationScreen*.
- **EventScreenController**: According to the button pressed, change the screen to *MainScreen* or *EventFirstCreationScreen*. It calls also the database to store the information needed to display inside the model.
- **EventFirstCreationScreenController**: According to the button pressed, change the screen to *MainScreen* or *EventFirstSecondScreen*. It calls also the database to store the information needed to display inside the model.
- **EventSecondCreationScreenController**:According to the button pressed, change the screen to *MainScreen* and validate the event's creation or *MainScreen* without any validation. It calls also the database to store the information needed to display inside the model.
- **EventModifyScreenController**: According to the button pressed, change the screen to *MainScreen* and validate the event's creation or *MainScreen* without any validation. Pressing *Modify* or *Delete* are changing an already existing event. It calls also the database to store the information needed to display inside the model.
- **TeacherEventController**: According to the button pressed, change the screen to *MainScreen* or *HomeworkScreen* or *CourseScreen*. It calls also the database to store the information needed to display inside the model.
- **CourseEventController**:According to the button pressed, change the screen to *MainScreen* and validate the event's creation or *MainScreen* without any validation. It calls also the database to store the information needed to display inside the model.
- **ModifyCourseController**: According to the button pressed, change the screen to *MainScreen* and validate the event's creation or *MainScreen* without any validation. Pressing *Modify* or *Delete* are changing an already existing event. It calls also the database to store the information needed to display inside the model.

- **HomeworkController**: According to the button pressed, change the screen to $MainScreen$ and validate the event's creation or $MainScreen$ without any validation. It calls also the database to store the information needed to display inside the model.
- **ModifyHomeworkController**: According to the button pressed, change the screen to $MainScreen$ and validate the event's creation or $MainScreen$ without any validation. Pressing $Modify$ or $Delete$ are changing an already existing event. It calls also the database to store the information needed to display inside the model.

2.2.2 Class Diagram

Here is an overview of the class diagram. It represents the link between the Model, View and Controller. On the next pages you will find a larger version of each Model, View and Controller

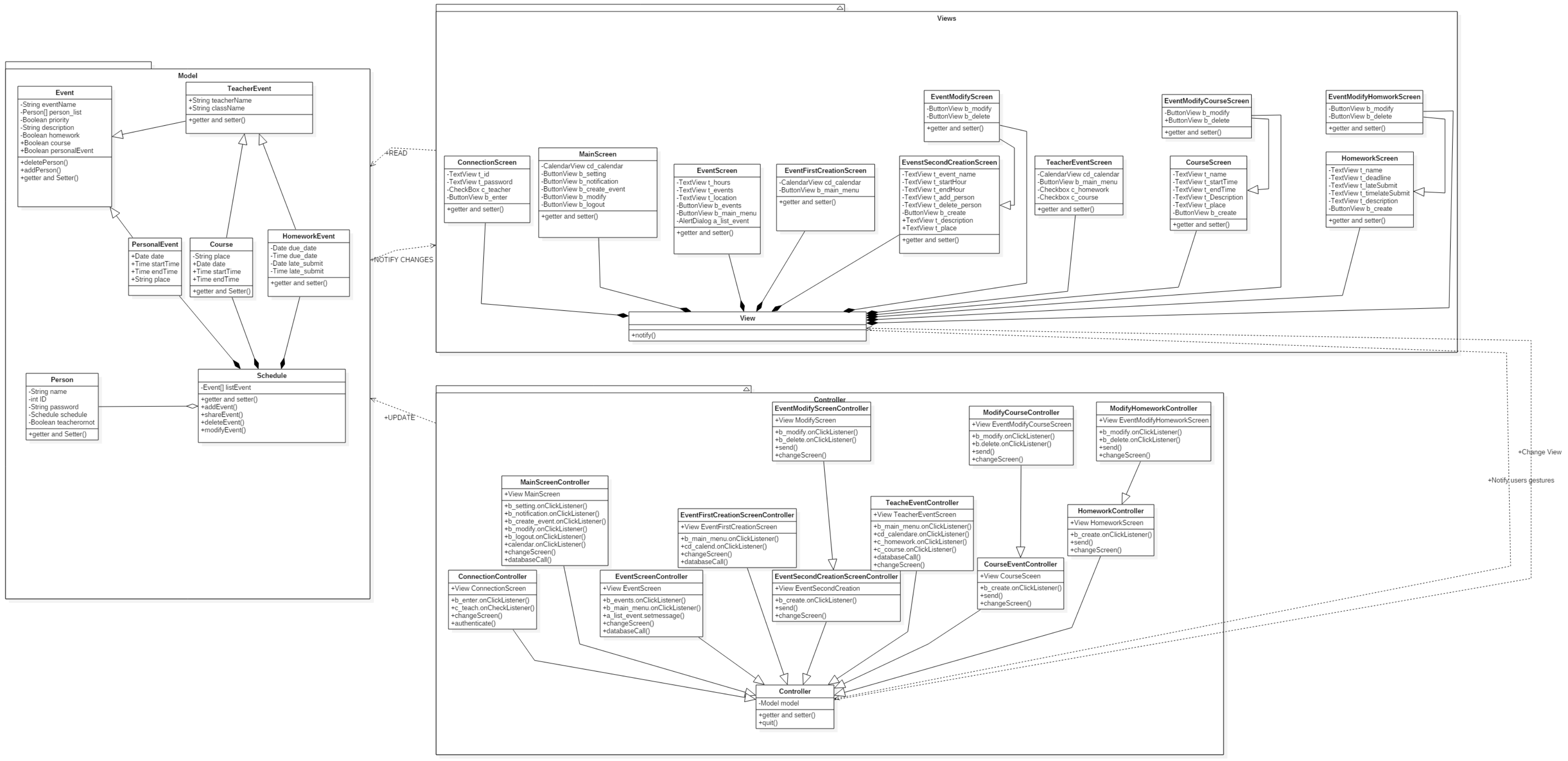


Figure 2: Overview of the classDiagram

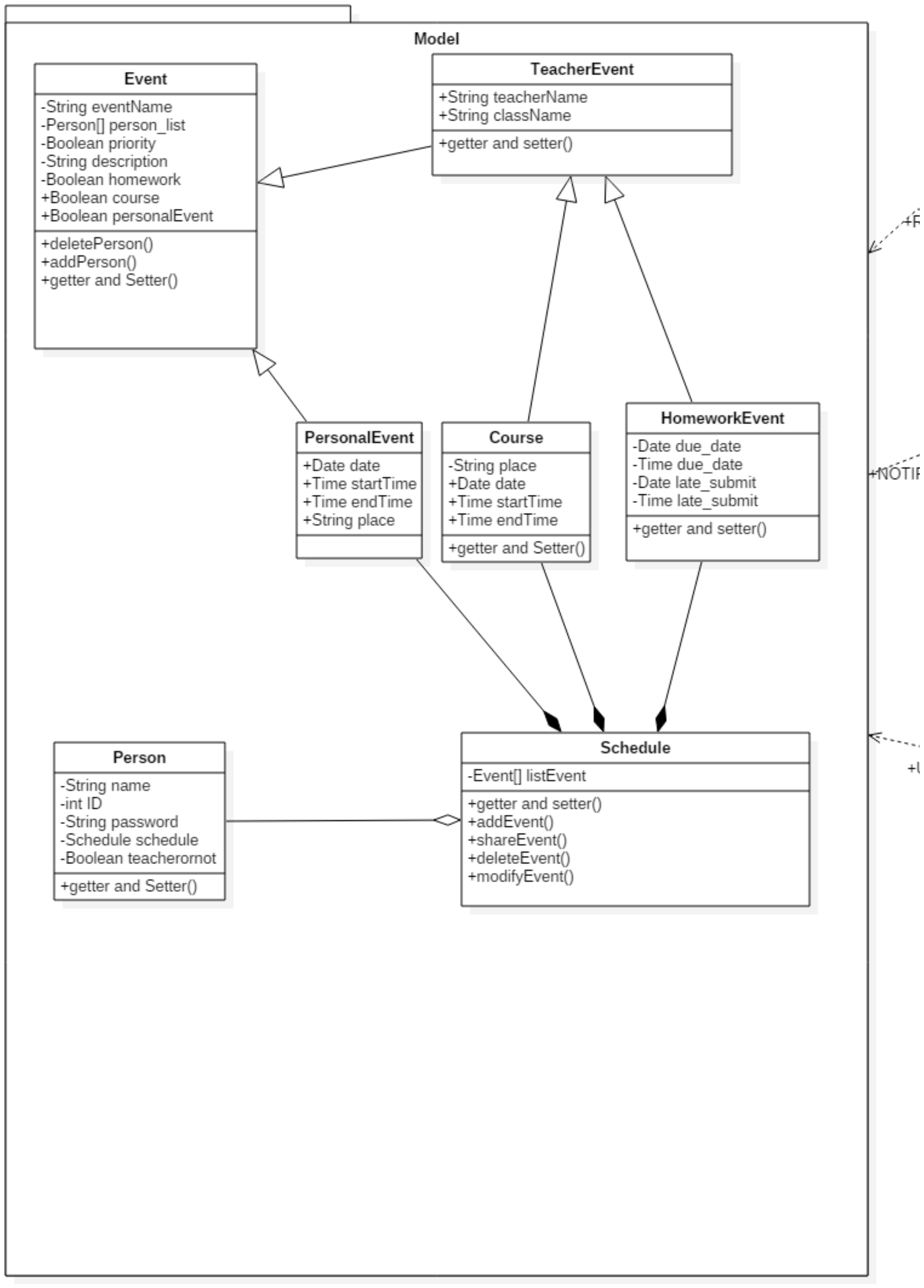


Figure 3: Model

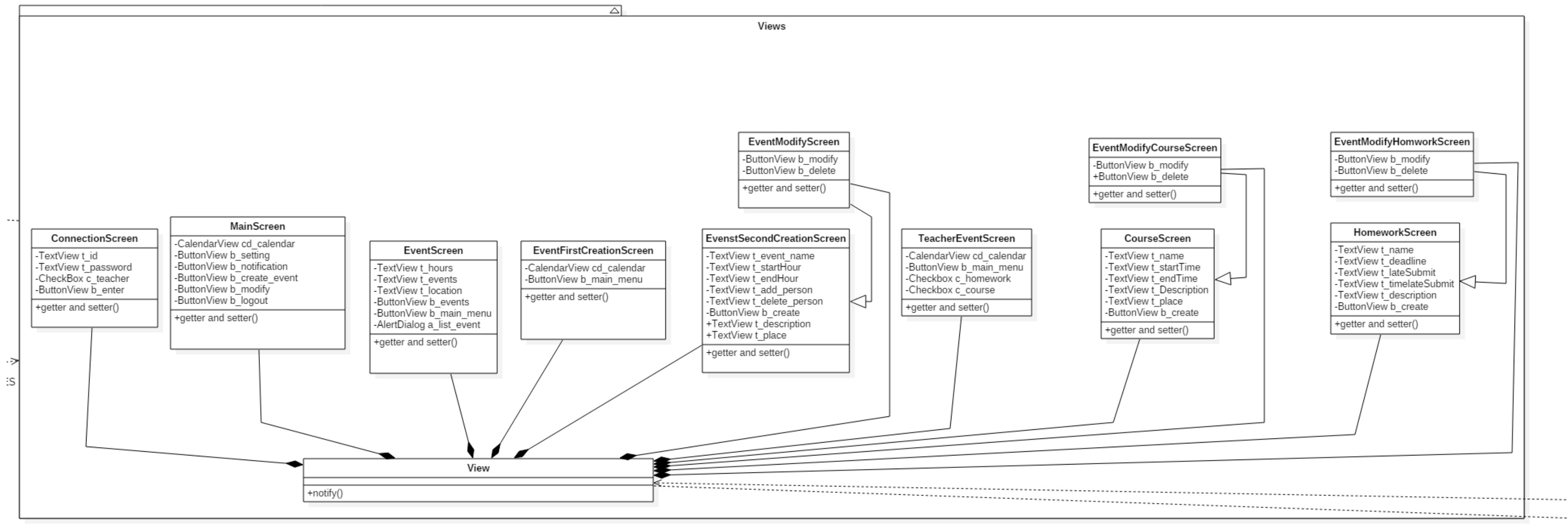


Figure 4: View

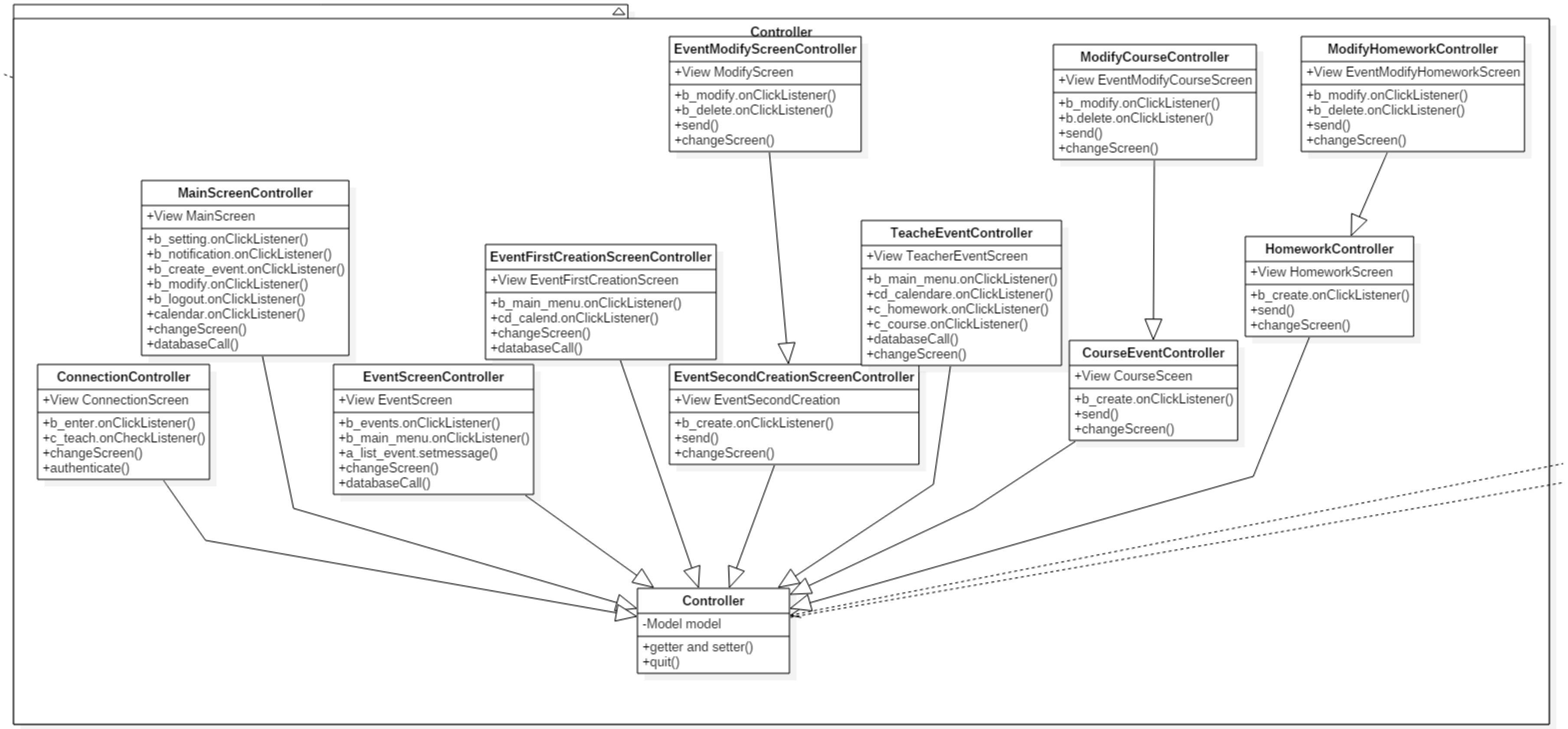

Controller
EventModifyScreenController
+View ModifyScreen
+b_modify.onClickListener()
+b_delete.onClickListener()
+send()
+changeScreen()
ModifyCourseController
+View EventModifyCourseScreen
+b_modify.onClickListener()
+b.delete.onClickListener()
+send()
+changeScreen()
ModifyHomeworkController
+View EventModifyHomeworkScreen
+b_modify.onClickListener()
+b_delete.onClickListener()
+send()
+changeScreen()
MainScreenController
+View MainScreen
+b_setting.onClickListener()
+b_notification.onClickListener()
+b_create_event.onClickListener()
+b_modify.onClickListener()
+b_logout.onClickListener()
+calendar.onClickListener()
+changeScreen()
+databaseCall()
EventFirstCreationScreenController
+View EventFirstCreationScreen
+b_main_menu.onClickListener()
+cd_calend.onClickListener()
+changeScreen()
+databaseCall()
TeacheEventController
+View TeacherEventScreen
+b_main_menu.onClickListener()
+cd_calendare.onClickListener()
+c_homework.onClickListener()
+c_course.onClickListener()
+databaseCall()
+changeScreen()
HomeworkController
+View HomeworkScreen
+b_create.onClickListener()
+send()
+changeScreen()
ConnectionController
+View ConnectionScreen
+b_enter.onClickListener()
+c_teach.onCheckListener()
+changeScreen()
+authenticate()
EventScreenController
+View EventScreen
+b_events.onClickListener()
+b_main_menu.onClickListener()
+a_list_event.setmessage()
+changeScreen()
+databaseCall()
EventSecondCreationScreenController
+View EventSecondCreation
+b_create.onClickListener()
+send()
+changeScreen()
CourseEventController
+View CourseSceen
+b_create.onClickListener()
+send()
+changeScreen()
Controller
-Model model
+getter and setter()
+quit()

Figure 5: Controller

2.3 User Interface Design

User Interface and interractions are more described in the Appendix 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.

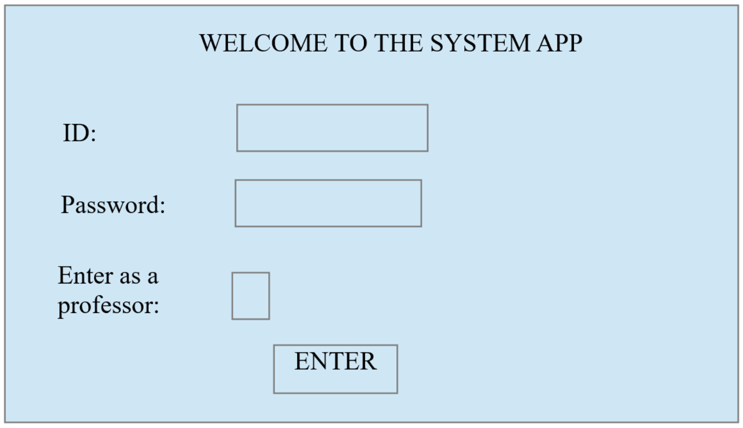


Figure 6: Connection Screen

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.

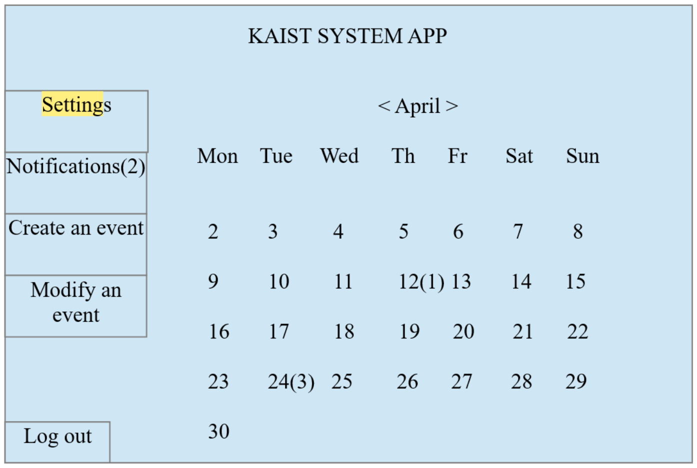
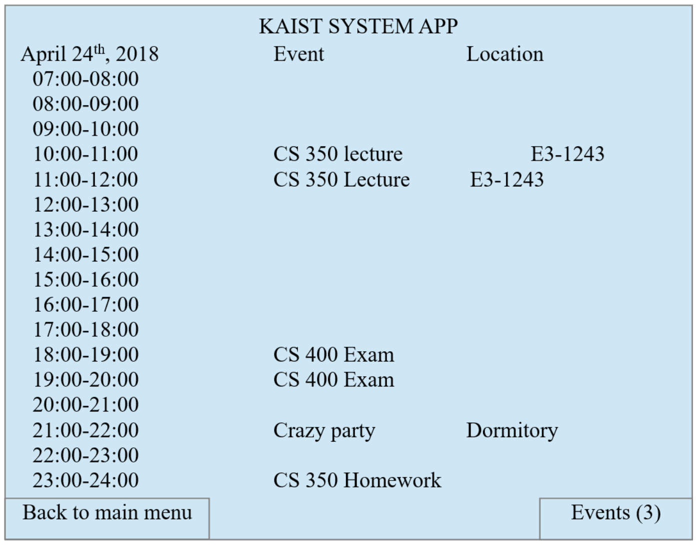


Figure 7: Main Screen



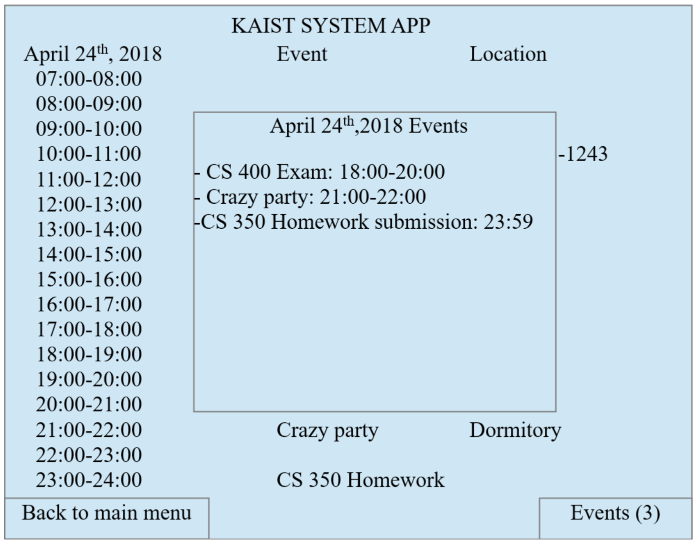
KAIST SYSTEM APP

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 8: 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.



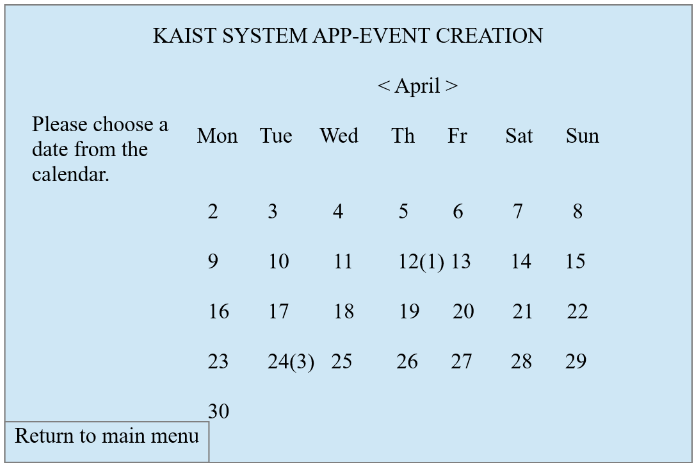
KAIST SYSTEM APP

April 24th, 2018	Event	Location
07:00-08:00		
08:00-09:00		
09:00-10:00		
10:00-11:00		-1243
11:00-12:00		
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		
19:00-20:00		
20:00-21:00		
21:00-22:00	Crazy party	Dormitory
22:00-23:00		
23:00-24:00	CS 350 Homework	

April 24th,2018 Events

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

Figure 9: Event Screen


KAIST SYSTEM APP-EVENT CREATION

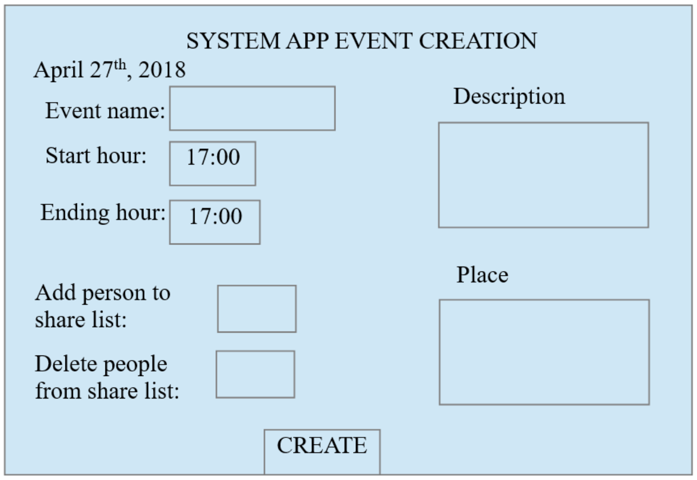
< April >

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 10: 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:

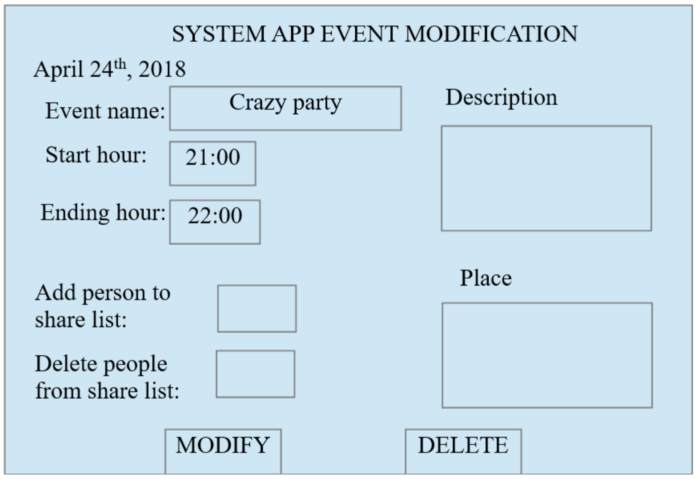
Delete people from share list:

Description

Place

CREATE

Figure 11: 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:

Description

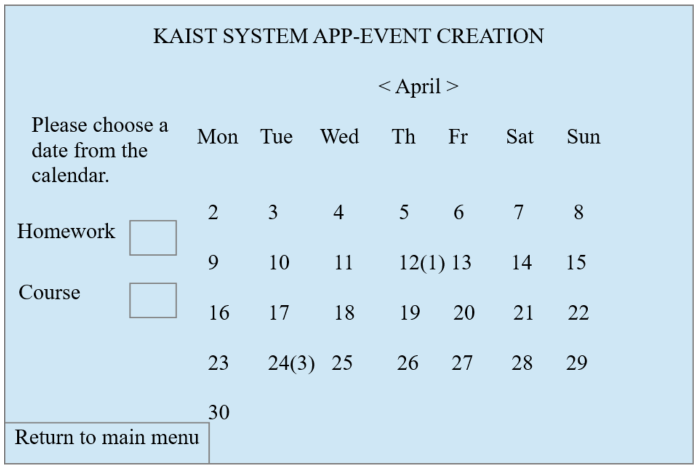
Place

MODIFY

DELETE

Figure 12: 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.


KAIST SYSTEM APP-EVENT CREATION
< April >
Please choose a date from the calendar.
Homework
Course
Return to main menu

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 13: Event Creation First Screen

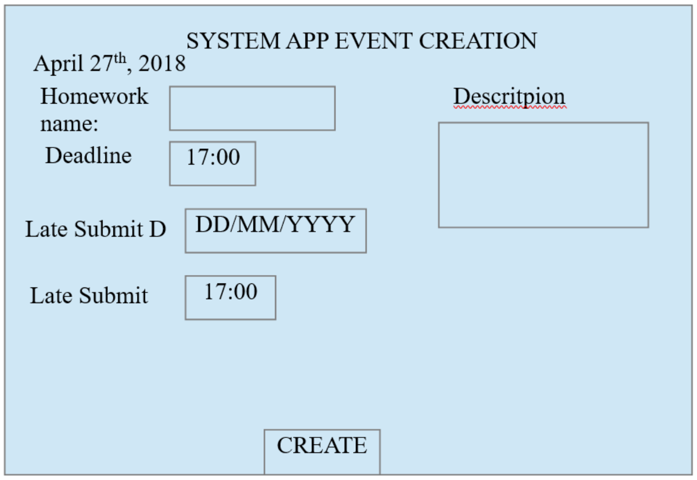


Figure 14: Homework Creation Second Screen

When a teacher is connected, the event creation will be slightly different. As you can see on **Figure 8**, there are two check-boxes, Homework and Course that appeared. The teacher can choose what he wants to add.
Figure 9 and **Figure 10** shows user interface for both Homework and Course creation event. The only differences with student interface are the options available.

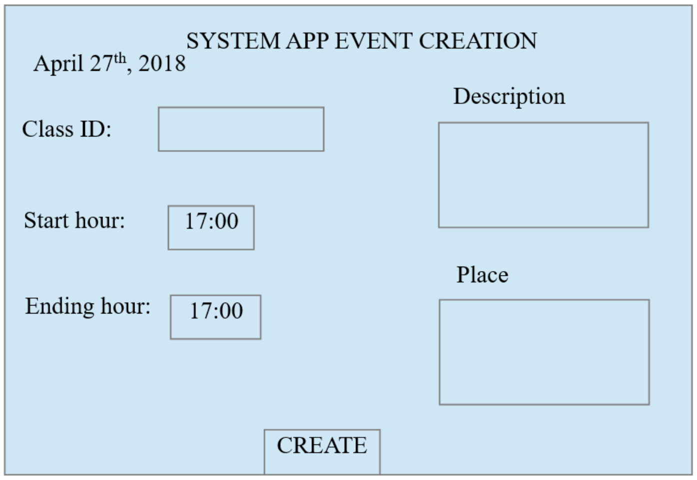
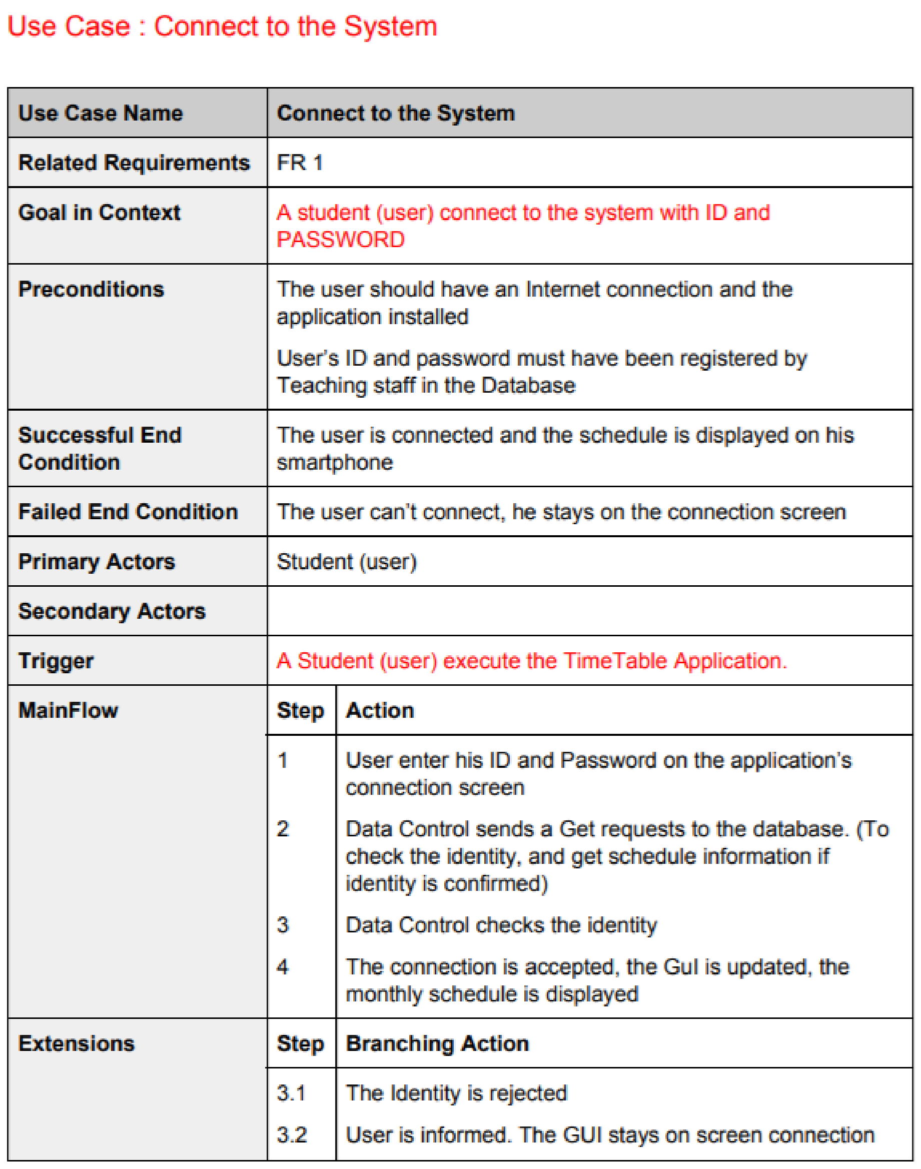


Figure 15: Course Creation Third Screen

2.4 (Refined) Use Case Diagram & Description

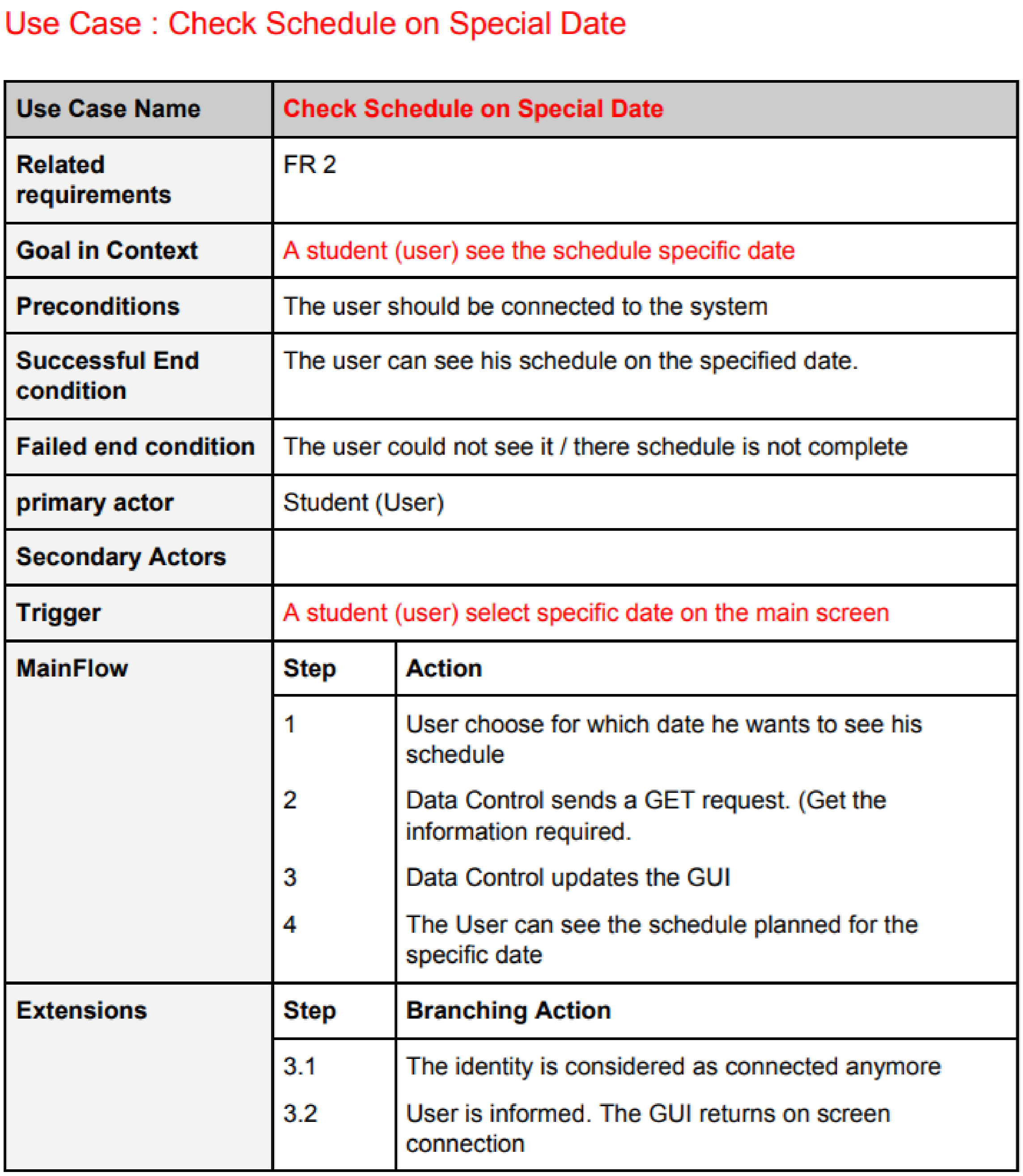
You will first find the refined use case description, and in the end of this part the new use case diagram.



Use Case : Connect to the System

Use Case Name	Connect to the System	
Related Requirements	FR 1	
Goal in Context	A student (user) connect to the system with ID and PASSWORD	
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	Student (user)	
Secondary Actors		
Trigger	A Student (user) execute the TimeTable Application.	
MainFlow	**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

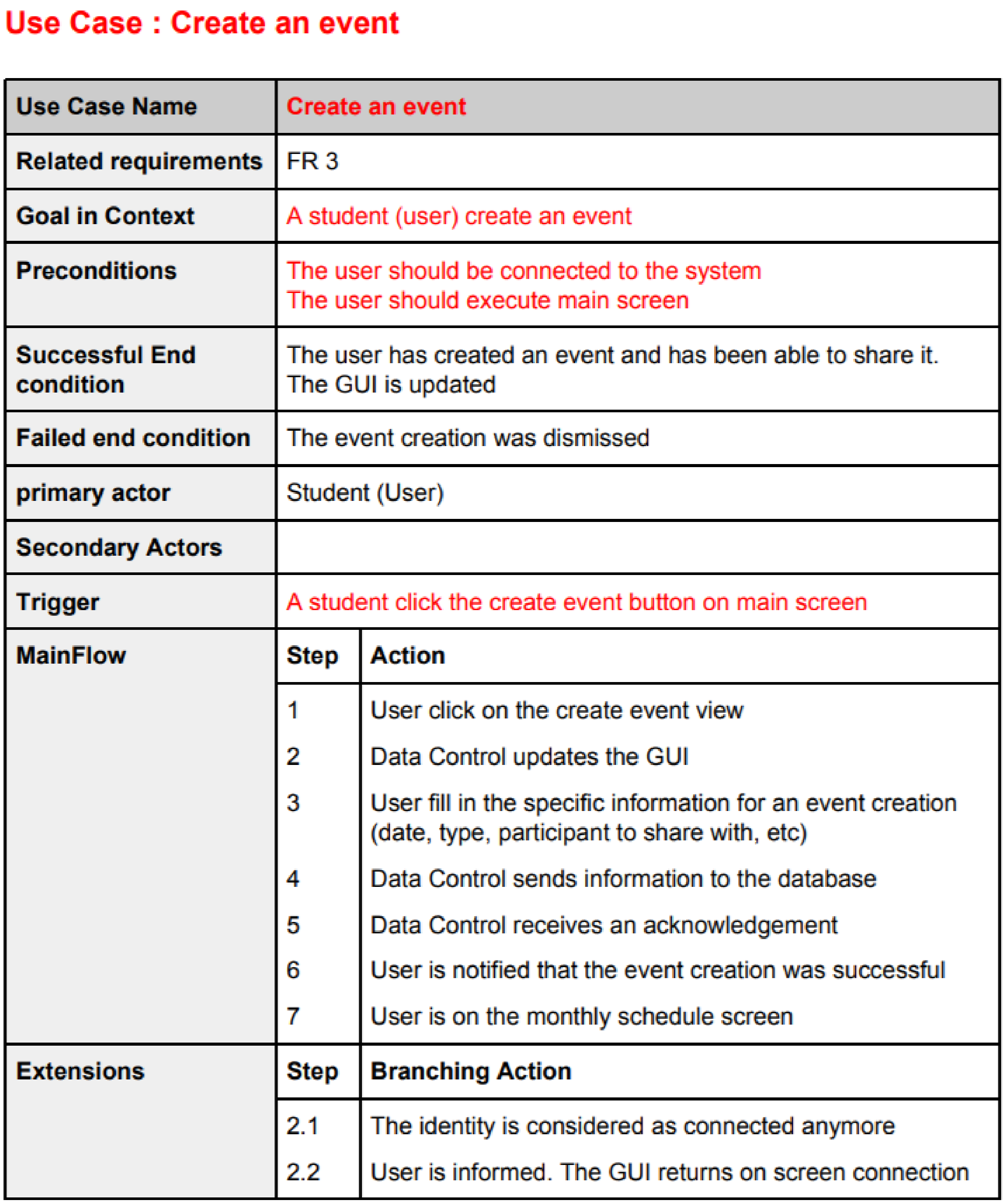
Figure 16: Description of Usecase: Create an event



Use Case : Check Schedule on Special Date

Use Case Name	Check Schedule on Special Date	
Related requirements	FR 2	
Goal in Context	A student (user) see the schedule specific date	
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 actor	Student (User)	
Secondary Actors		
Trigger	A student (user) select specific date on the main screen	
MainFlow	**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

Figure 17: Description of Usecase: Connect to the system and Check Schedule on a special date



Use Case : Create an event

Use Case Name	Create an event	
Related requirements	FR 3	
Goal in Context	A student (user) create an event	
Preconditions	The user should be connected to the system The user should execute main screen	
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 actor	Student (User)	
Secondary Actors		
Trigger	A student click the create event button on main screen	
MainFlow	**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 acknowledgement
	6	User is notified that the event creation was successful
	7	User is on the monthly schedule screen
Extensions	**Step**	**Branching Action**
	2.1	The identity is considered as connected anymore
	2.2	User is informed. The GUI returns on screen connection

Figure 18: Description of Usecase: Connect to the system and Check Schedule on a special date

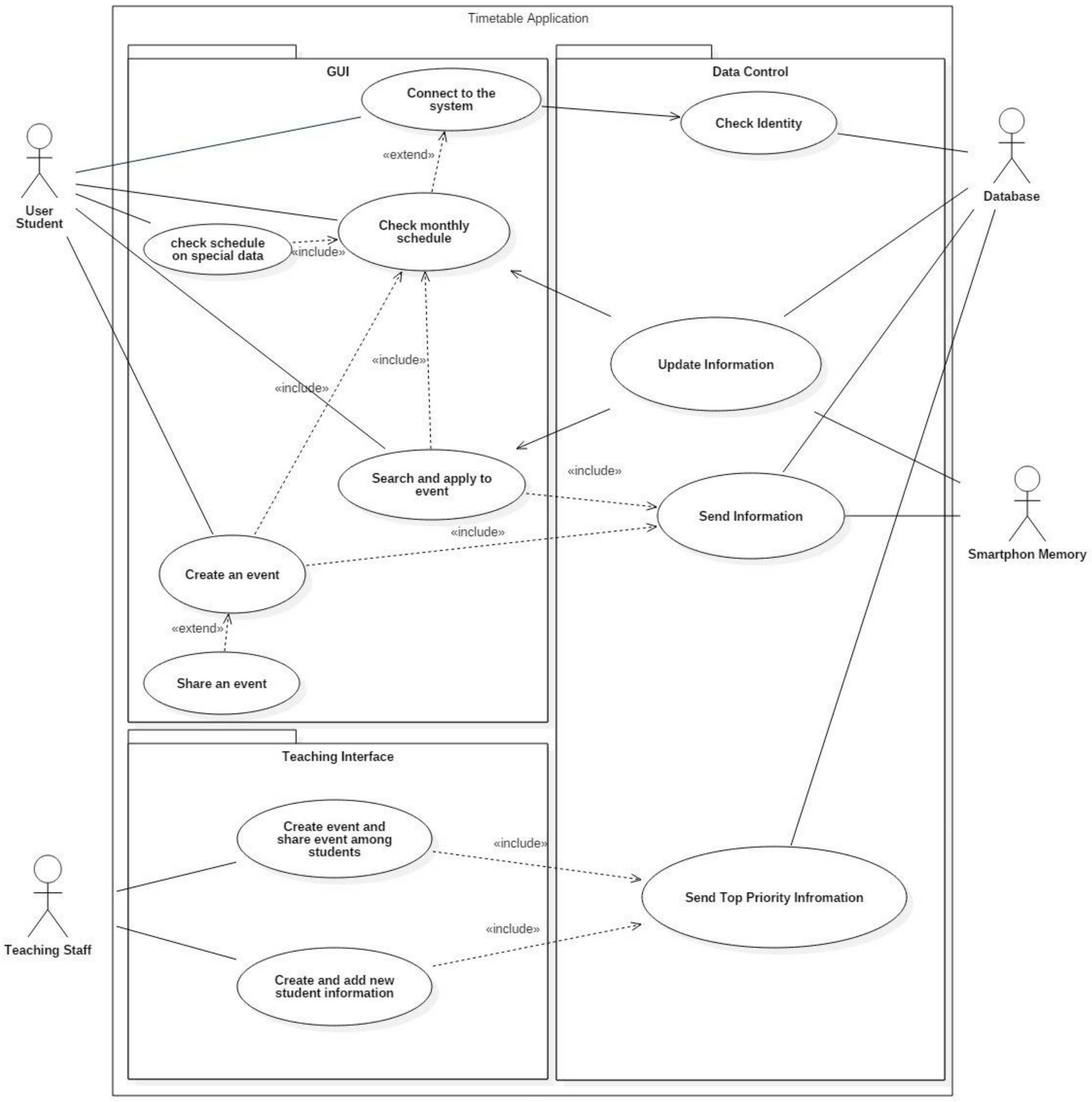


Figure 19: Usecase diagram refined

2.5 (Refined) Sequence Diagram

Here you will find an improved version of the sequence diagram from project #2.

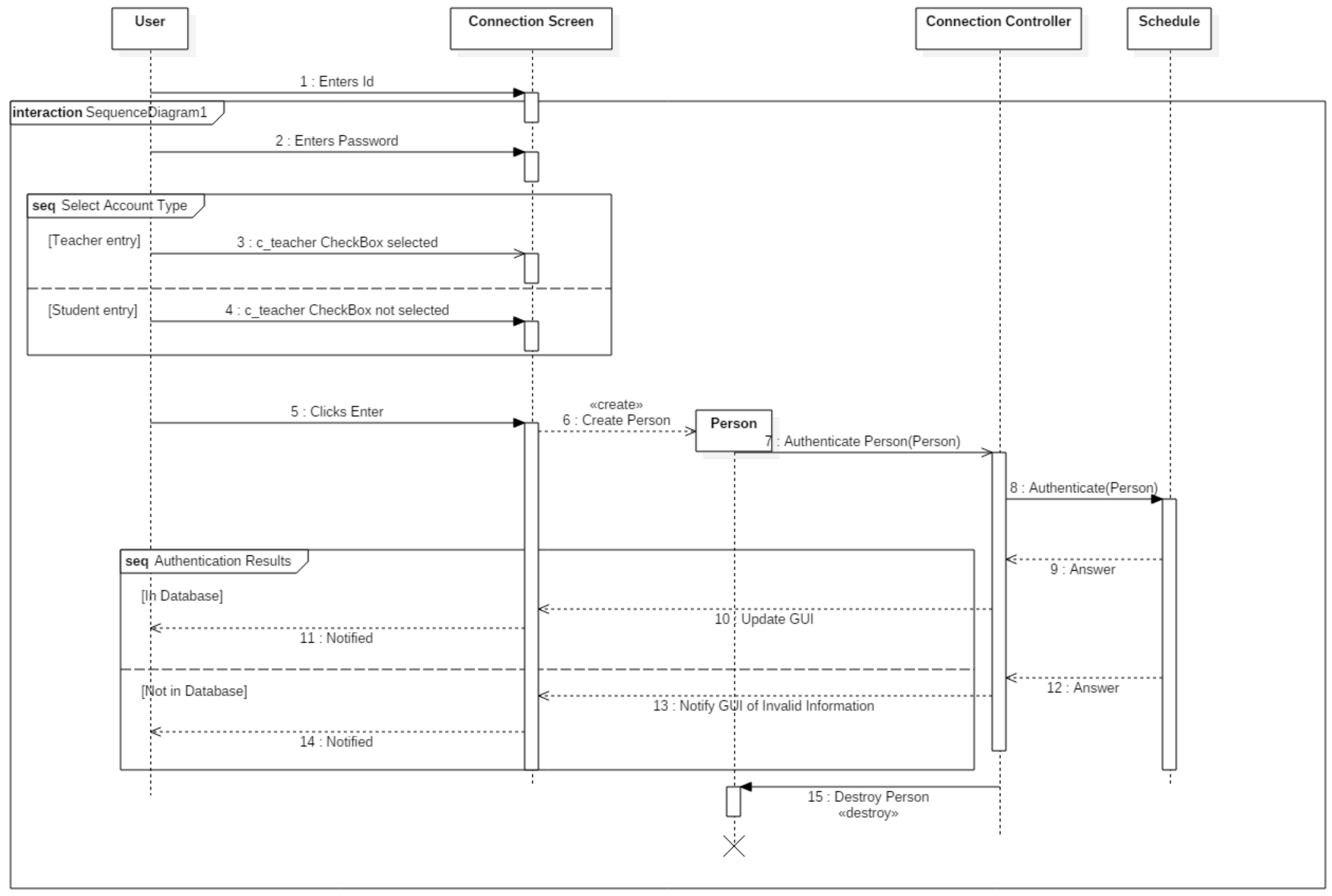


Figure 20: Sequence Diagram #1

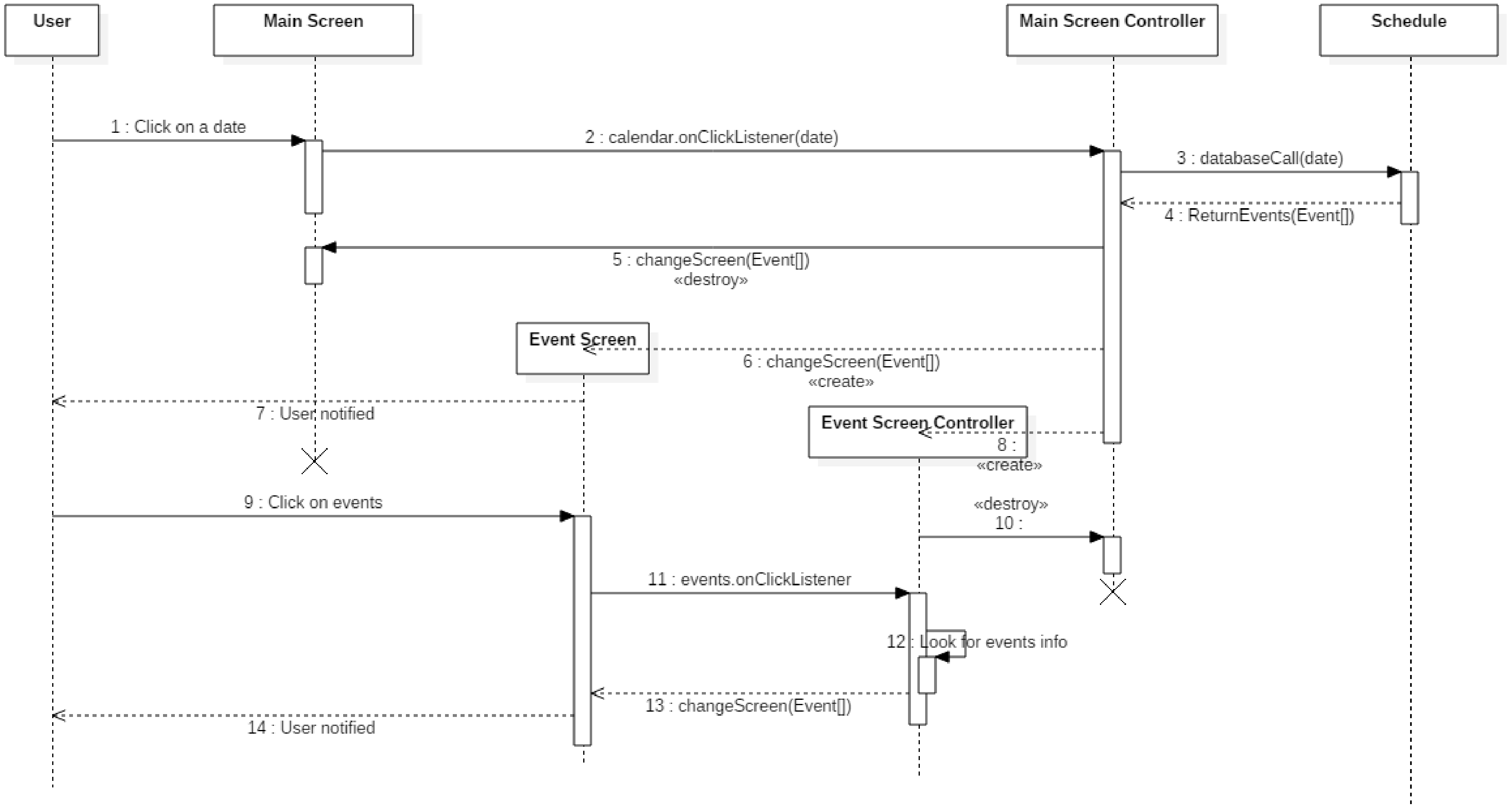


Figure 21: Sequence Diagram #2

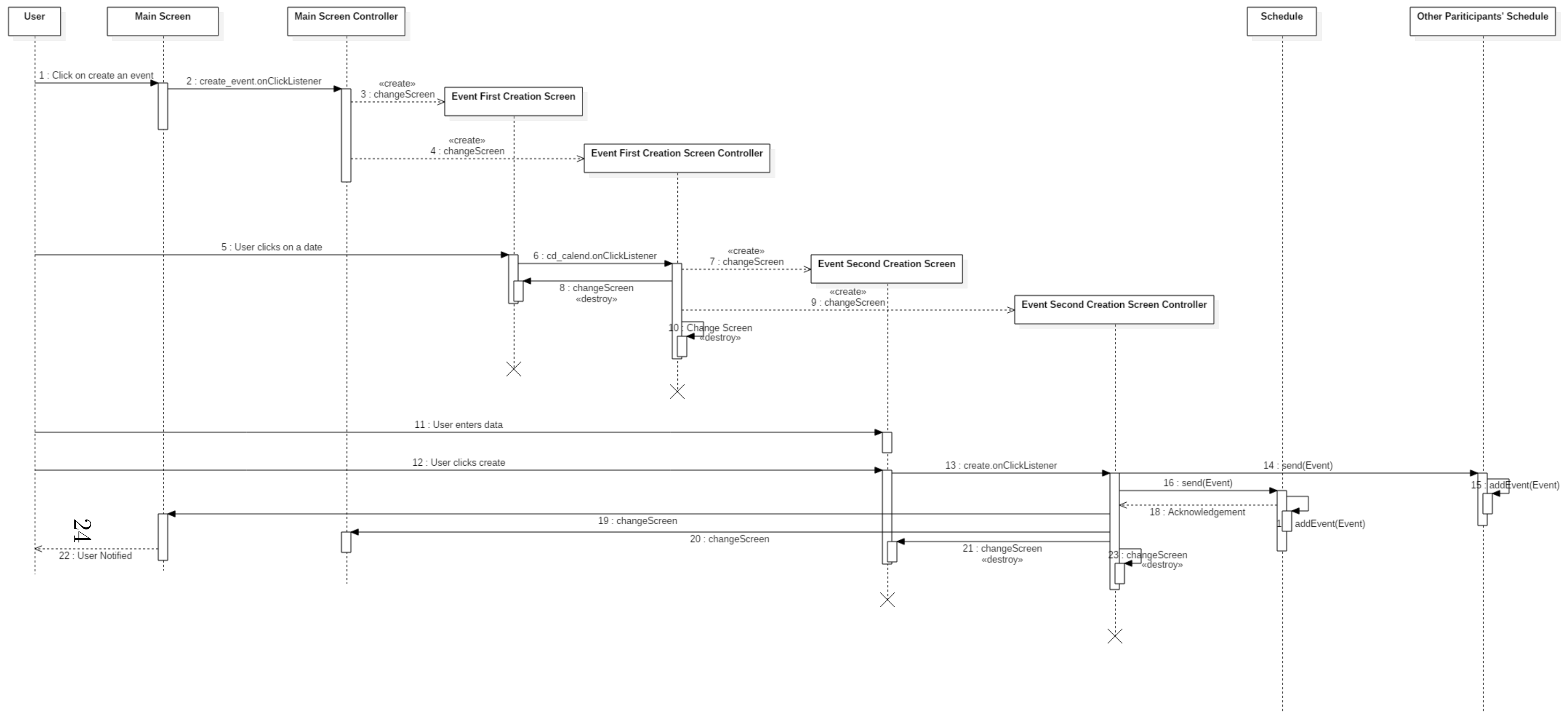
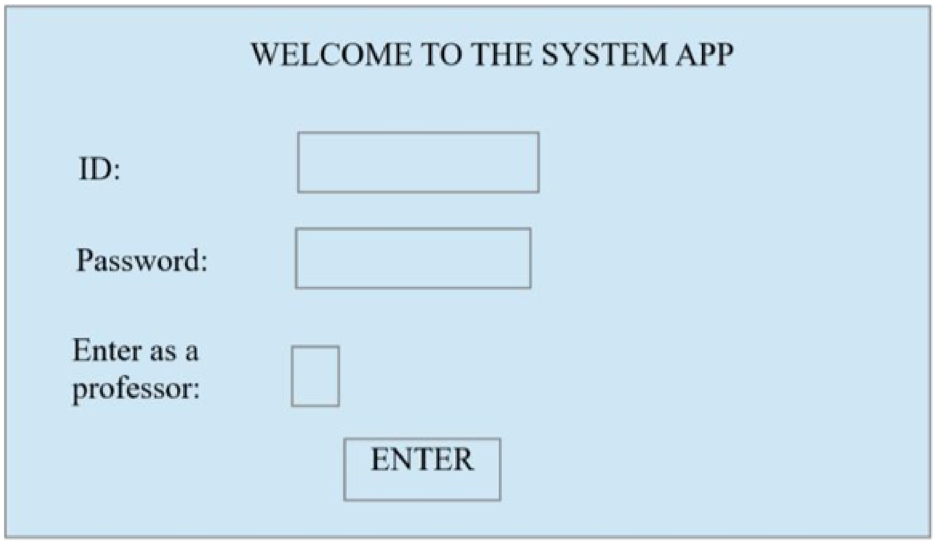


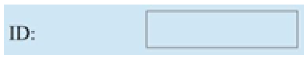
Figure 22: Sequence Diagram #1

3 Appendix

3.1 Connection View

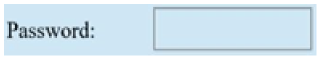


1. ID Insert Space :

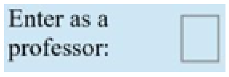
   - Put an ID in square box.

2. Password Insert Space

   - Put a Password in square box.
   - Password view type is ***. ex) 12345678 → ********

3. Check Box of ‘Enter as a professor’

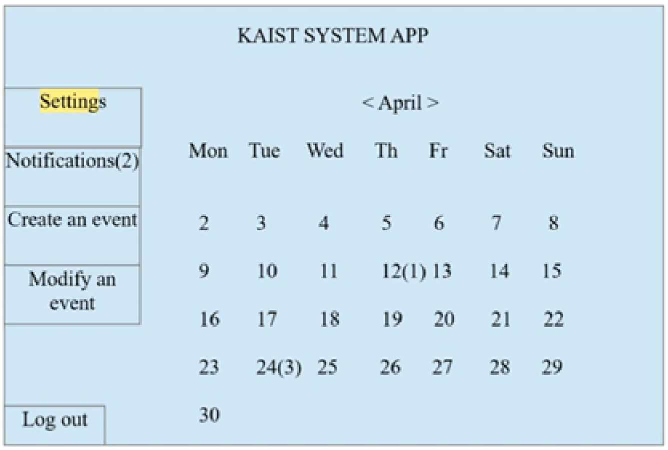
   

   - By clicking the box, you can check and uncheck

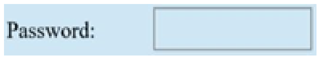
4. Enter

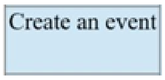
   - Depending on the correct ID and Password, it is decided to go main screen when the button is pressed.

3.2 Main View



1. Setting Button

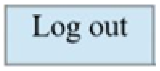
   - Click Setting Button for changing language, font, color.
2. Notification Button
   

   - Click Notification Button for the following two things :
     - Denote the number of new events that are supposed to take place today.
     - Provide warning message if there are multiple events at the same day, same hour.
3. Create an event Button
   

   - Click Create an event Button for Event Creation View.
4. Modify an event Button

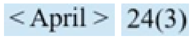
   - Modify an event by clicking the respective button.
   - Event Modification View

5. Logout Button



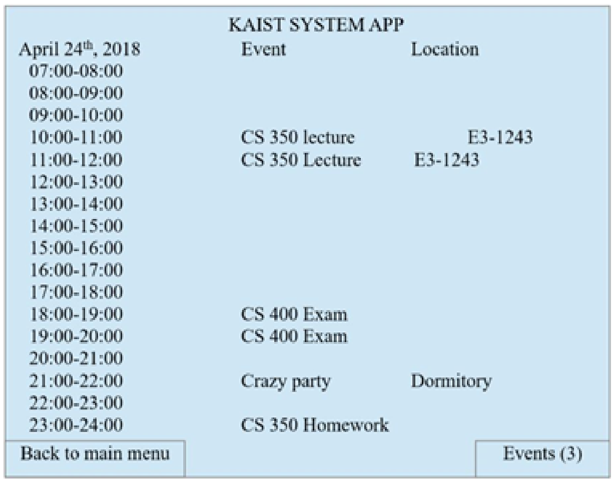
    - For logout

6. Date Button



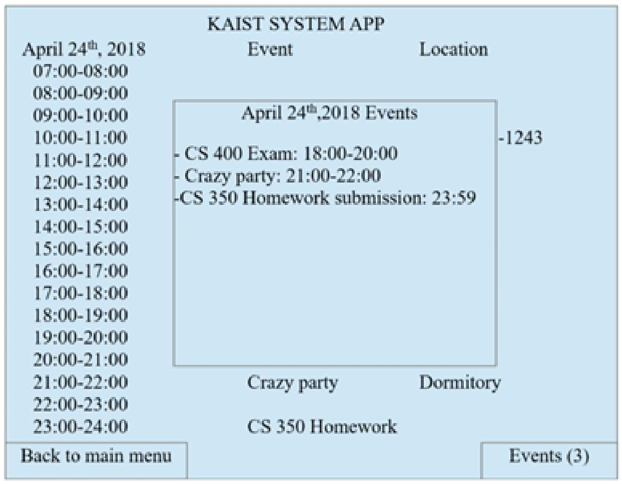
    - Press the “<” or “>” signals next to the name of the month for changing month.
    - Day(number of events)
    - Click date for an Event View

3.3 Event View



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	



KAIST SYSTEM APP

April 24th, 2018	Event	Location
07:00-08:00		
08:00-09:00		
09:00-10:00		
10:00-11:00		-1243
11:00-12:00		
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		
19:00-20:00		
20:00-21:00		
21:00-22:00	Crazy party	Dormitory
22:00-23:00		
23:00-24:00	CS 350 Homework	

April 24th,2018 Events

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

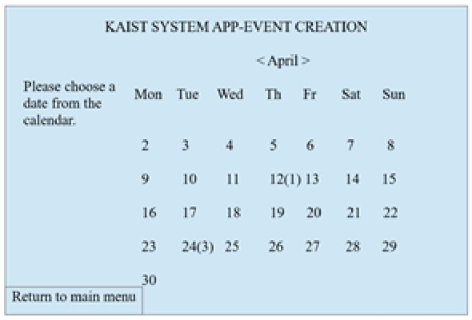
1. Back to main menu Button

   - Click Button for moving main menu

2. Events Button

   - There is two type of Event View (Figure 1, Figure 2)
   - Change view type of Event View

3.4 Event Creation View

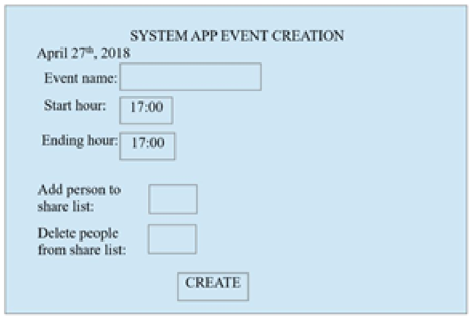

KAIST SYSTEM APP-EVENT CREATION

< 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


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

1. Date Button

   < April > 24(3)

   - Press the “<” or “>” signals next to the name of the month for changing month.
   - Day(number of events)
   - Click date for an Event View

2. Back to main menu Button

   Back to main menu

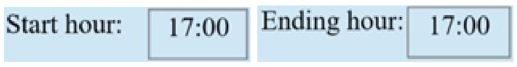
   - Click Button for moving main menu

3. Event name Insert Space

   Event name:

   - Put event name in square box

4. Start hour Select Space, Ending hour Select Space

   
   Start hour: 17:00 Ending hour: 17:00

   - Click on the square box and select the desired time from the list

5. Add person to share list

   Add person to share list: Delete people from share list:

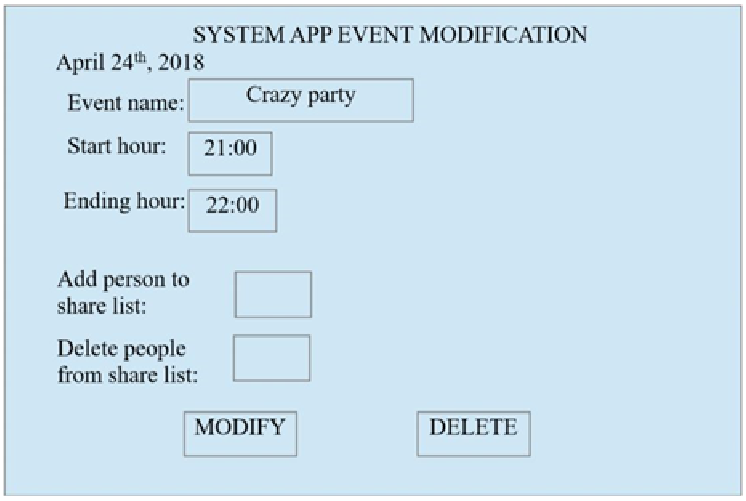
   - Enter the name of the person you want to delete or add

6. Create Button

   CREATE

   - When all the settings are completed, click the button to create a new event.

3.5 Event Modification View


SYSTEM APP EVENT MODIFICATION

April 24th, 2018

Event name: Crazy party

Start hour: 21:00

Ending hour: 22:00

Add person to share list:

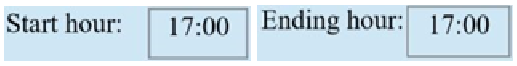
Delete people from share list:

1. Event name Insert Space

Event name:

- Put event name in square box

2. Start hour Select Space, Ending hour Select Space


Start hour: 17:00

Ending hour: 17:00

- Click on the square box and select the desired time from the list

3. Modify Button

- When all modifications are completed, click the button to create a new event

4. Delete Button

- Click the button to delete the event

4 Acknoledgements

Editor of this document : Alexandre Allani

Redactors:

- System Overview : Kuntae Park
- System Architecture : All the team, created by Shin Seung Hun, commented by Alexandre Allani
- Class Diagram : Alexandre Allani
- User Interface Design : From SRS, modified by Alexandre Allani & Shin Seung Hun
- (Refined) Use Case Diagram & Description : Shin Seung Hun
- (Refined) Sequence Diagram : Bilgehan Bingol
- Appendix : Shin Seung Hun