Software Requirements Specification

Version 1.2

for

Skynet Room Reservation System

Prepared by

Gabriele Bavaro	27399103	gabriele.bavaro@bell.net
Tian Li Zhou	27407637	Zhou_tian-li@hotmail.com
El-Mehdi Beghdadi	26781276	Elmehdi.beghdadi@google.com
Emili Vasseva	27526741	emvasseva@gmail.com
Dias Marat	27277911	
Bruce Edouard Brazier	27419562	
Sean Marcoux	27511876	

Instructor: *Dr C. Constantinides*

Course: SOEN 342

Date: November 23, 2016

Document history

Date	Version	Description	Author
October 1, 2016	1.0	Functionalities and Risks added	Gabriele Bavaro El-Mehdi Beghdadi Tian Li Zhou
October 1, 2016	1.01	Create tentative on Domain Model	Gabriele Bavaro El-Mehdi Beghdadi Tian Li Zhou
October 16, 2016	1.02	Wrote Use Case makeReservation And make system software diagrams for it, communication diagram, system operations document and contracts.	Gabriele Bavaro El-Mehdi Beghdadi Tian Li Zhou
October 23, 2016	1.03	Added functional and non-functional requirements to this document	Gabriele Bavaro El-Mehdi Beghdadi Tian Li Zhou
October 24, 2016	1.04	Edit re-write certain parts of my section (nf requirements).	Gabriele Bavaro El-Mehdi Beghdadi Tian Li Zhou
November 1, 2016	1.1	Rough finished copy of SRS document completed	Gabriele Bavaro El-Mehdi Beghdadi Tian Li Zhou
November 19, 2016	1.2	Modified SRS document, added appendix, final review	Gabriele Bavaro Tian Li Zhou Medhi-El Beghdadi Sean Marcoux Emili Vasseva Bruce Edouard-Brazier

			Dias Marat

Table of contents

1. Introduction 5

Purpose 5

Scope 5

Definitions, acronyms, and abbreviations 5

References 6

2. Overall description 7

Product perspective 7

Product functions 7

User characteristics 8

3. Specific requirements 9

External interfaces 9

Functionality 11

Actor goal list 11

Use case view 12

Reliability 14

Maintainability 14

Portability 15

Design constraints 15

(On-line) user documentation and help 15

Purchased components 15

Licensing requirements 15

Legal, copyright and other notices 16

3. Analysis Models 17

4. Analysis Models 17

List of figures

Figure 1. Use case model. 12

1. Introduction

Purpose

The purpose of the Software Requirements Specifications (SRS) is to define and communicate the many and different software requirements for the product Room Reservation. The version of the product that the SRS document deals with is version 1.0. The audience of the SRS document are the stakeholders of the Room Reservation product and the information contained within the SRS, particularly the requirements, are to be documented in such a way as to provide an understanding to the stakeholders on what the requirements are. The structure of this SRS is inspired by the I-EEE standard 830-1998[1].

Scope

Room Reservation is an online conference room reservation system software product that reserves different rooms from different buildings, for a set period of time, for use by a student or a group of students which are the end users. The online software system allows students to reserve rooms for a given time slot and to reserve multiple time slots for the same room or different rooms up to a certain limit of rooms and/or timeslots. In addition, if a timeslot for a room is already taken a student may opt to be placed on a waiting list for that room and timeslot. The system only allows authorized users to access and manipulate it. The product is not an open system. The beneficiaries of the software product are the students who as the end users utilize it and Concordia University, which is the stakeholder. The product is developed for Concordia University, a university based in Montreal, Quebec.

Definitions, acronyms, and abbreviations

FR Functional Requirements

FTP File Transfer Protocol

HTTP Hypertext Transfer Protocol

MYSQL My Structured Query Language

NFR Non-Functional Requirements

OO Object Oriented

SRS Software Requirements Specifications

References

[1] IEEE std 830-1998, IEEE Recommended Practice for Software Requirements Specifications (SRS), IEEE Computer Society.

[2] Dr. C. Constantinides, Software Architecture and Design I Term Project, Concordia University, September 13, 2016.

2. Overall description

Product perspective

The purpose of the application is the same as an existing software from Concordia University. It is use to book room reservations for Capstone projects. However, both softwares are not connected to each other. Different database are also used. The software is a mobile application version and it is self-contained.

Product functions

The required high-level features and functions are:

1) Create reservations:
    a) Add student to selected available time slot of a specific room;
    b) A maximum of 3 reservations per student per week;
2) Remove reservations.
    a) Remove student from selected reserved time slot;
3) Modify reservations.
    a) Remove student from selected reserved time slot and add student to the new selected time slot;

Other features and functions are:

1) Display reservations:
    a) Display a weekly calendar (current week);
    b) Display schedule with rooms including their available and reserved time slots;
2) Add/remove to waiting list:
    a) Add or remove student on the waiting list of the selected reserved time slot;
    b) A maximum of 3 waiting lists per student per week;
3) Send notifications;
    a) Send a reminder to student 1 hour before the reservation time slot;
    b) Send a notification to the student who successfully reserve a time slot;
        i) From directly reserving.

ii) From waiting list

c) Notify the current position of the student on the waiting list

4) Reset weekly reservations.

a) Each week, all time slots of the rooms are reset to empty.

5) Remove student from other waiting lists

a) Remove student from other waiting lists that are at the same time as the successful reserved time slots;

User characteristics

The targeted users of this application are registered engineering students of Concordia University. The users must have applied to a Capstone project in order to use the application for reserving rooms.

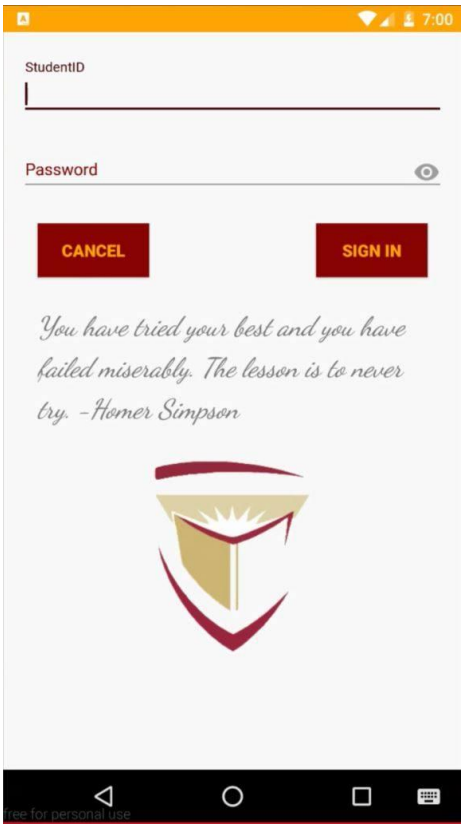
Constraints, assumptions and dependencies

1. The system need to be Object-Oriented.
2. An Android smartphone shall be used as platform.
3. Android version 5.0 + shall be used.
4. JDK version 1.8 + shall be used;
5. The application will be developed in English, as well as its artefacts (code and comments, deployment scripts, and unit tests);
6. Spring Framework shall be used to develop the android application.

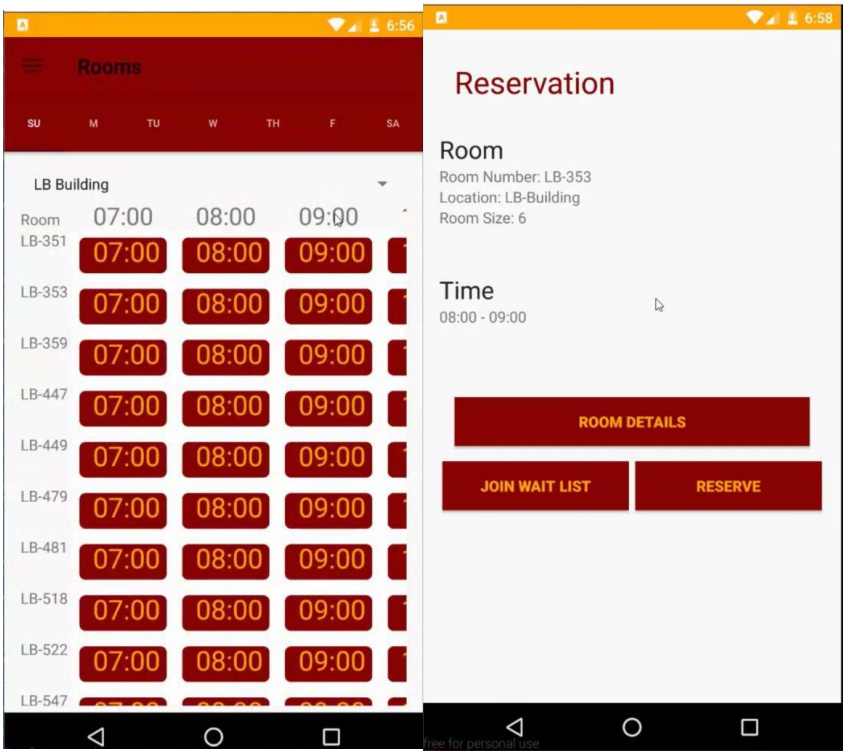
3. Specific requirements

External interfaces

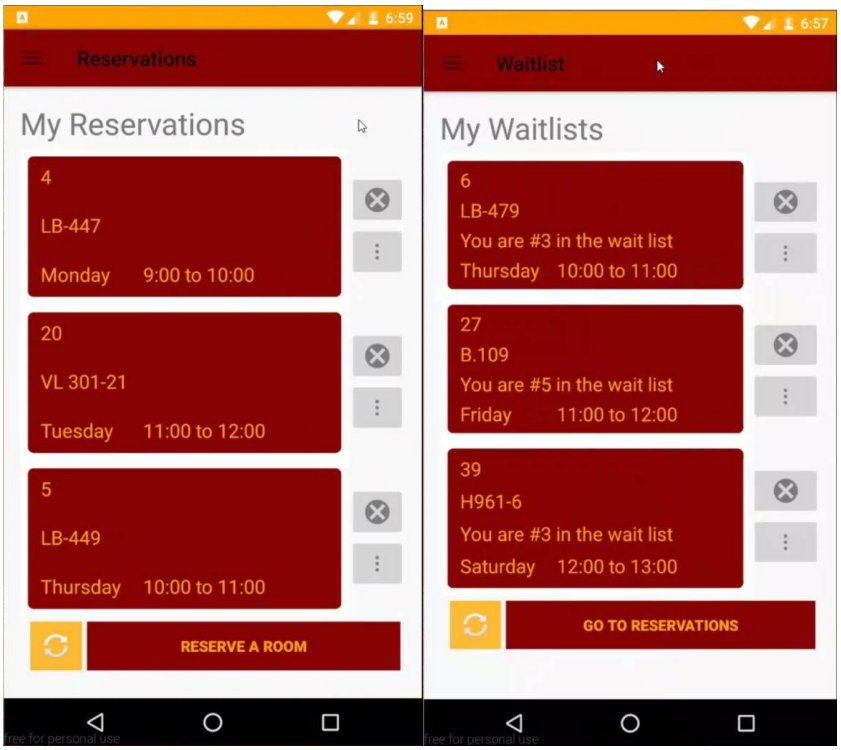
3.1.1 User interface



The user shall be able to authenticate into the application by entering his student ID and his password.



The application shall have a page that display a room x time scheduler in which contains multiple timeslots. The user will be able to click on a specific timeslot to put himself in the list (reserving or put in the waitlist).



The user shall have a list of his reservations and a list of his wait list. The user will be able to click on the delete button to cancel the reservation. The user will also be able to click on the modify button to directly choose an new timeslot to replace.

3.1.2 Hardware Interfaces

The application shall be used with an android smartphone. The smartphone is the only hardware required in order to use the application. The smartphone needs to support Android version 5.0 or higher.

3.1.3 Software interface

The system shall be connected to a MYSQL database in which it contains the information of the students, the rooms, and reservations. The application shall also be connected with the Spring framework which is used to develop the application.

3.1.4 Communication interfaces

No specific web browsers are required since it is an android mobile application.

Functionality

This section contains the Actor Goal List which represent the functional requirements and its actors. The list capture the intended behaviour of the system. This section also contains the Use Case view.

Actor goal list

Actor	**Goal**
Student	Authentication
	View reservations
	Create a reservation

	Cancel a reservation
	Modify a reservation.
	Add to waiting list
	Remove from waiting list

Use case view

The use case model is shown in Figure 1.

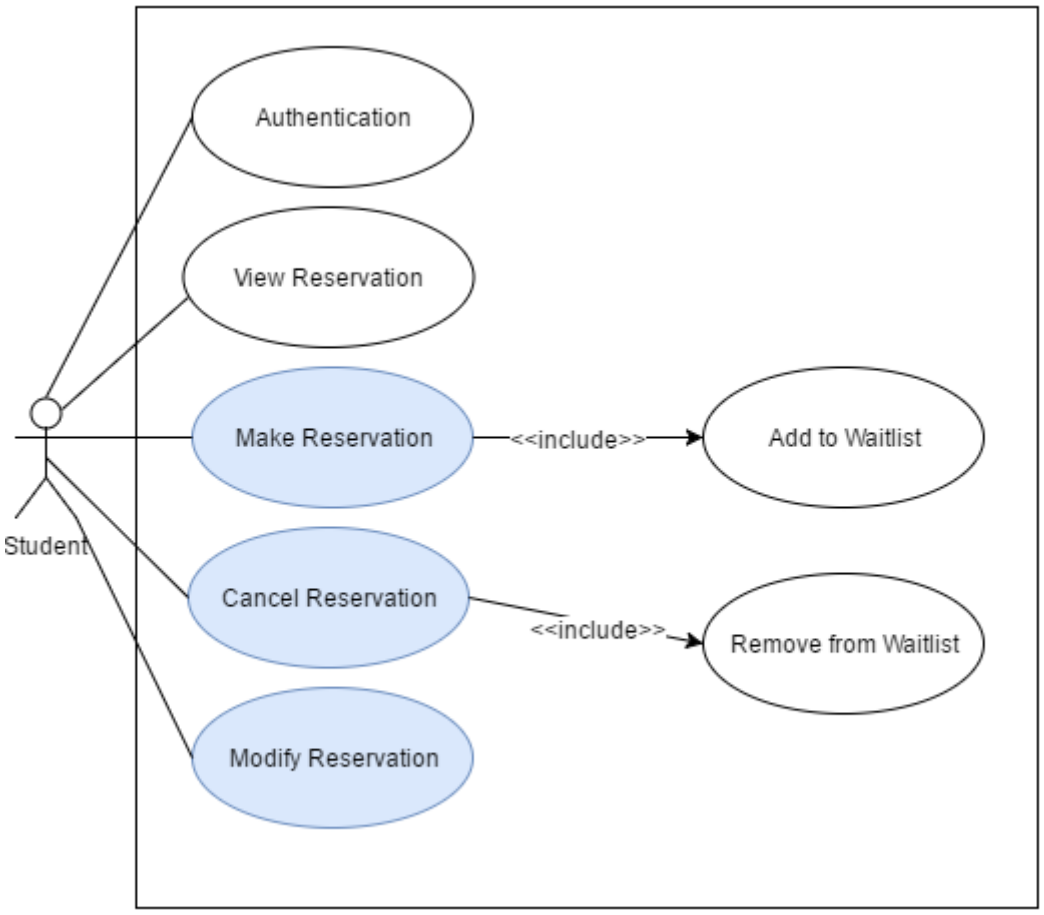


Figure 1. Use case model.

Functional Suitability

- **Functional completeness:** the system shall offer all functional requirements that are deemed critical (login,create, cancel reservations and add to waitlist) which represent 87% of the functional requirements and as many of the functional requirements mentioned in the above section as possible.
- **Functional appropriateness:** Each user activity (make, replace or cancel reservation) shall not take more than 3 steps for the user to accomplish.

Performance efficiency

- **Time behaviour:** The response to each user click or touch screen tap should take less than 2 seconds.
- **Resource utilization:** On the front-end, the user shall use an android mobile device with version 4.0 or 4.1 installed on it to be able to run the application. The system back-end will use a Wamp Server which consists of an Apache web server, MySQL database which will be connected to the Spring framework (Java language).

Usability

- **Appropriateness recognizability:** 80% of users should find that the system satisfies their needs.
- **Learnability:** It should take less than 30 minutes for a new users to figures out how to add, change and cancel reservations.
- **Operability:** It should take less than 3 clicks (taps on touch screen) for the user to accomplish any of the main activities (add reservation, add to waitlist, change reservation or cancel reservation).
- **User error protection:** The system shall send reminder messages 1h before reservation time starts so student can cancel them if they can't make it. CRUD operations shall require confirmation at the end for the changes performed to be saved in the database.
- **User interface aesthetics:** UI should implement many interface patterns found in similar applications to reduce confusion. A small sample survey shall be used to test it.
- **Accessibility:** User experience should feel familiar in its implementation and UI to 90% of users. Color blindness shall be taken into consideration when using colors to indicate results of operations. Front-end design shall follow Android best practices to allow augmentation of font size by user if necessary.

Compatibility

- **Co-existence.** The Android front-end mobile application shall co-exist with the Spring backend framework.

- **Interoperability.** The Android front-end mobile application shall exchange and communicate and receive information from the backend Spring framework through GET and POST HTTP requests.

Reliability

Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time. This characteristic is composed of the following sub-characteristics:

- **Availability**. The system shall maintain a directory of rooms and their availabilities at different time slot. A Room instance shall only be accessed by one user at a time for the operations reserve/cancel/update.

  The system needs to have an algorithm in place to help resolve the conflict of two users arriving at the exact same time to reserve a room.

Security

- **Confidentiality.** The system shall not disclose the identity of the room holders who have confirmed reservations to other users nor the identity of the people on the waiting list.
- **Integrity.** The system shall be safe and fair to every user.
- **Non-repudiation.** Database transactions shall be logged and saved.
- **Accountability.** Logs shall not be modifiable by Administrator.
- **Authenticity.** Long password (minimum 8 characters) shall be required from users.

Maintainability

- **Modularity.** A multi-layered system shall be designed to separate responsibilities and lower coupling. An object-oriented architectural style shall be used.
- **Reusability.** Multi-layer architecture shall allow main domain classes to be reused if need be as they do not directly communicate with low level layers.
- **Analysability.** Logiscope shall be used to analyse the code. The report produced for this characteristic will include analysis of weighted methods per class, class comment rate, number of base classes and direct classes associated to each class. The resulting grade shall not be below fair.
- **Testability.** Logiscope shall be used to analyse the code. The report produced for this characteristic will include analysis of weighted methods per class, the total number of methods per class and the number of classes used directly by each current class. The resulting grade shall not be below fair.

Portability

- **Adaptability.** Android app shall work for devices with an Android OS version of 4.0 or 4.1. User interface shall automatically fit different screen sizes of users' mobile devices that run Android OS 4.0 or 4.1.

Design constraints

The design constraints that need to be followed are that the product must be an online system that utilizes an OO programming language alongside any libraries that can be incorporated into said language. The system may also be a mobile app as well, such as an android app. In addition the system must implement a database that will be incorporated into the finished product. In addition the system must support multiple users accessing and using it without any software bugs or performance issues showing up. The finished product must be constructed using an online system constructing framework, an example would be Android Studios. However the framework cannot do automatic work such as automatically implementing databases. The code structure must include structural and behavioural patterns. No purchased components were needed for the product.

(On-line) user documentation and help

The on-line documentation that a user can use to better understand the initial product will be developed for a later version of the software system.

Purchased components

No components were purchased to construct the product. All components used were either open sourced or free licenses.

Licensing requirements

No products used to construct the system needed to be licensed for a monetary value. All the tools licensed to construct the system had free licenses.Geny motion (an emulator) and Intellija (backend code developer) have free licenses which were used for the project. Those tools

which were used but did not have any licenses associated with them were Android Studio (front end code developer), Spring (framework) and MYSQL(database).

Legal, copyright and other notices

All products used to create the product were free online tools which were legally downloaded from their respective websites. Some of the products utilized did have licenses associated with them. A list of these is found in the Licensing Requirements section. The rest of the products used to construct the final system were open source products and as such did not have any licenses or copyrights associated with them.

3. Analysis Models

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS's requirements.

Illustrate (system) ***UML sequence diagrams*** (one for each critical scenario), identify system operations and describe operation contracts, one per critical system operation. You may also use ***UML state diagrams*** to describe critical use cases. Additionally, create a **domain model** for the system. Make sure that each model is traceable to the requirements.

4. Analysis Models

4.1- Create a reservation (critical)

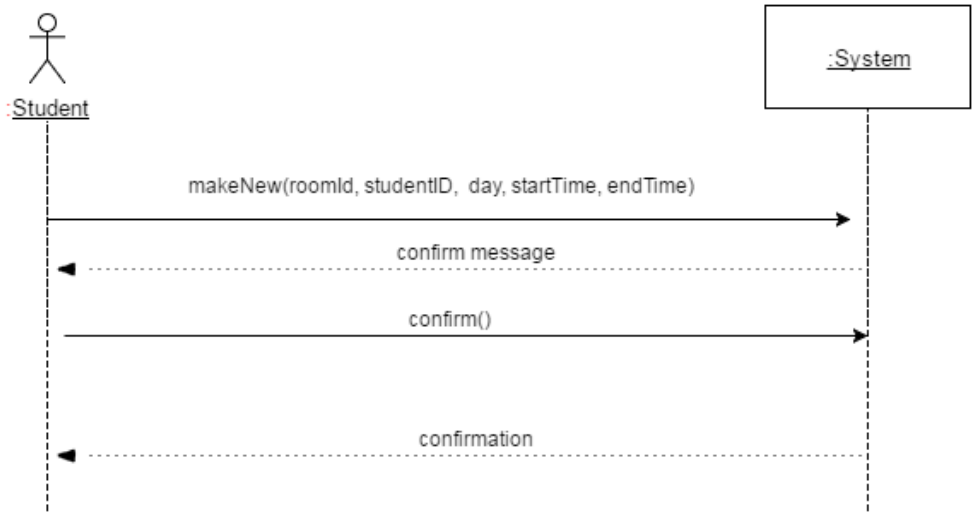
The user shall be able to create a reservation, and receive a confirmation at the end. A fully dressed use case follows for this critical requirement.

Use case UC1	Make Reservation
Primary Actor	Student
Stakeholders and Interests	Student: Wants to reserve a room
Preconditions	- Student is identified and authenticated - Student has at least one reservation left from the maximum of 3
Success Guarantee (Postconditions)	- A Reservation instance was created and includes the student and the room for the specified timeslot reserved.
Main success scenario (or basic flow)	1 - Student requests to reserve a room. 2 - System display confirmation message.

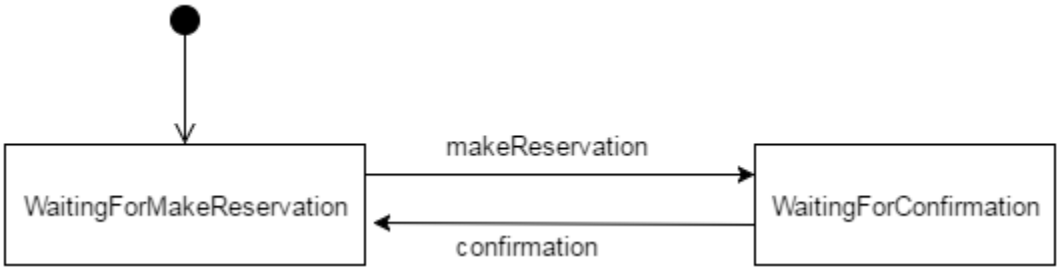
Extensions (or alternative flows)	- If student has used all of his/her 3 reservations, system will indicate an error message. -If selected room is reserved at desired time, system offers to be added to a waitlist. -If student has already reserved 3 times, and they want to reserve another free room, the system offers the student the option to replace one of his/her current reservations with the previously selected one.
Special Requirements	Touch screen UI
Technology and Data Variations List	Mobile app

UML system sequence diagram:

The actor first informs the system of the changes they require, only after they confirm it does the system apply the changes.



State Diagram:



System Operations:

System Operations:
makeNew(roomId, studentId, day, startTime, endTime)

Operation Contract:

Contract CO1	makeRoomReservation
Operation	makeNew(roomId, studentId, day, startTime, endTime)
Cross References	UC1: Make Reservation
Preconditions	- Student is identified and authenticated - Student has at least one reservation left from the maximum of 3
Postconditions	- A Reservation instance was created (instance creation)

4.2: Add to Waitlist(non-critical)

The user shall be able to add himself/herself to a wait list, and receive a confirmation at the end. A fully dressed use case follows for this requirement.

Use case UC4	Add to Waitlist
Primary Actor	Student
Stakeholders and Interests	Student: Student wants to be added to the waitlist of a room at a specific time.

Preconditions	- Student is identified and authenticated - Student has at least 1 from a total of 3 appended waitlists. - Room has to be already reserved at the desired time.
Success Guarantee (Postconditions)	- Number of left appended waitlists is decreased by 1. - Student is appended to the waitlist of a room at a specific time.
Main success scenario (or basic flow)	1 - Student requests to append to the waitlist of a room. 2- System display confirmation message.
Extensions (or alternative flows)	If students decides to append to a waitlist but has no more available “appended waitlist” slots, the system displays an error message.
Special Requirements	Touch screen UI
Technology and Data Variations List	Mobile app
Open issues	

4.3- Cancel a reservation (critical)

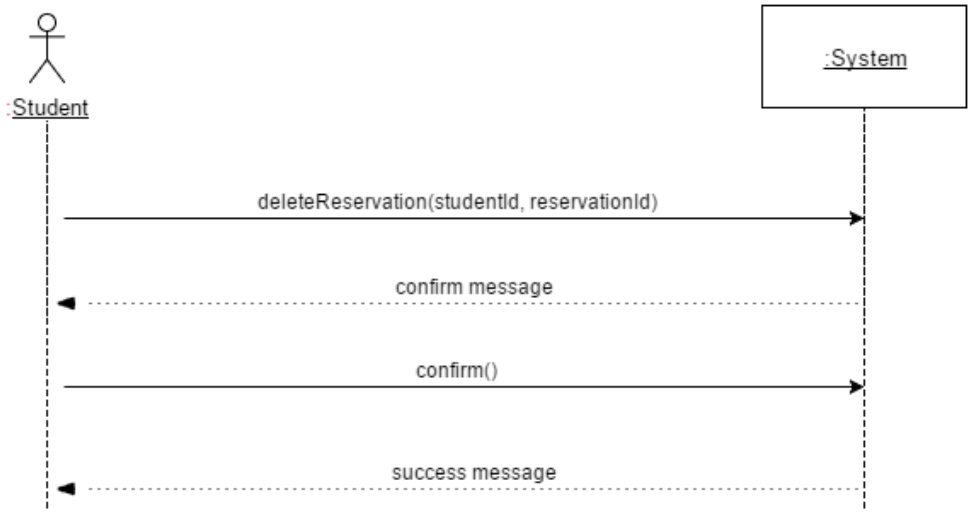
The user shall be able to cancel a reservation, and receive a confirmation at the end. A fully dressed use case follows for this critical requirement.

Use case UC2	Cancel Reservation
Primary Actor	Student
Stakeholders and Interests	Student: Wants to cancel reservation
Preconditions	- Student is identified and authenticated - Student has already reserved the room.
Success Guarantee (Postconditions)	- Reservation is deleted from the reservations list.

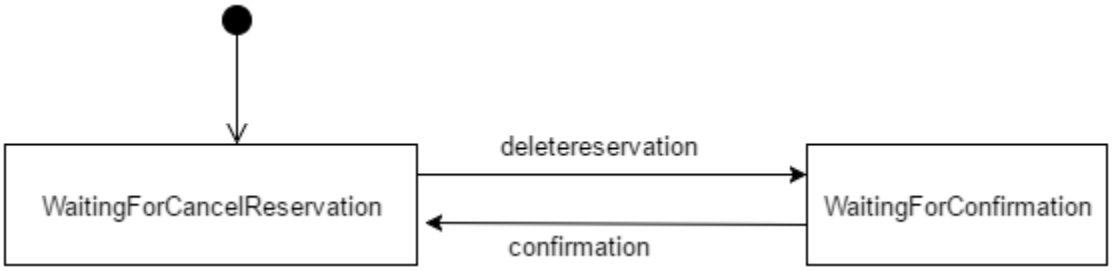
	- Number of reservations left to the student is incremented by one.
Main success scenario (or basic flow)	1 - Student requests to cancel his/her reservation. 2 - System display confirmation message. 3- Student confirm 4- System display success message
Extensions (or alternative flows)	
Special Requirements	Touch screen UI
Technology and Data Variations List	Mobile app
Open issues	

UML system sequence diagram:

The actor first informs the system of the changes they require, only after they confirm it does the system apply the changes.



State Diagram:



System Operations:

System Operations:
deleteReservation(studentId,reservationId)

Operation Contract:

Contract	Delete reservation
Operation	deleteReservation(studentId,reservationId)
Cross References	UC2: Cancel Reservation
Preconditions	- Student has been authenticated - Student has chosen the reservation to delete
Postconditions	- Reservation instance r was deleted. (instance deletion) - Attribute “position” of the reservations that has higher position number(lower in the waitlist) was decreased (Attribute modification)

4.4- Authentication (non-critical)

Use case UC7	Authentication
Primary Actor	Student
Stakeholders and Interests	Student: Students wants to login to the system.
Preconditions	- Student is accessing the console

Success Guarantee (Postconditions)	- Student is identified and authenticated
Main success scenario (or basic flow)	1 - Student accesses the console. 2 - System prompts the student to enter username and password. 3 - Student inputs a username and password. 4 - System displays a confirmation message.
Extensions (or alternative flows)	-
Special Requirements	Touch screen UI
Technology and Data Variations List	Mobile app
Open issues	

4.5- View Reservation (non-critical)

The user shall be able to login into the overall system and receive confirmation at the end. A fully dressed use case follows for this requirement.

Use case UC5	View Reservations
Primary Actor	Student
Stakeholders and Interests	Student: Students wants to see all reserved and available rooms
Preconditions	- Student is identified and authenticated
Success Guarantee (Postconditions)	-
Main success scenario (or basic flow)	1 - Student requests to search for available and reserved rooms. 2 - System prompts the student to choose a day of

	the week. 3 - Student picks a day. 4 - System displays a table of rooms x times with the available and not available reservations.
Extensions (or alternative flows)	-
Special Requirements	Touch screen UI
Technology and Data Variations List	Mobile app
Open issues	

4.6- Remove from a wait list (non-critical)

The user shall be able to remove himself/herself from a wait list, and receive a confirmation at the end. A fully dressed use case follows for this requirement.

Use case UC5	Remove from Waitlist
Primary Actor	Student
Stakeholders and Interests	Student: Student wants to be removed from a waitlist.
Preconditions	- Student is identified and authenticated - Student is already in the waitlist of the room/time he/she wants to get removed from.
Success Guarantee (Postconditions)	- Number of “appended waitlists” slots is increased by 1. - Student is removed from the waitlist.
Main success scenario (or basic flow)	1 - Student requests to cancel his/her reservation on a waitlist. 2 - System display confirmation message. 3- Student confirm

	4- System display success message
Extensions (or alternative flows)	-
Special Requirements	Touch screen UI
Technology and Data Variations List	Mobile app
Open issues	

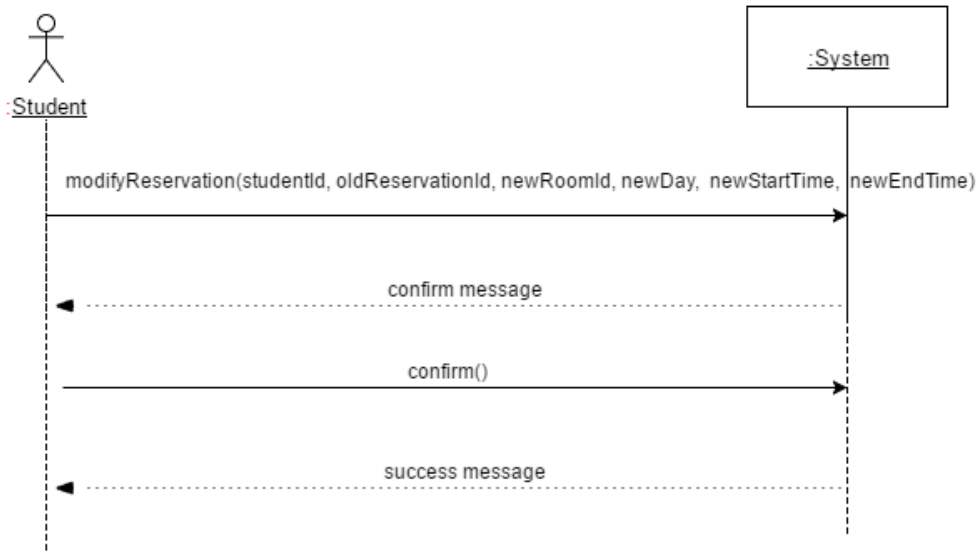
4.7- Modify a reservation (critical)

The user shall be able to modify a reservation, and receive a confirmation at the end. A fully dressed use case follows for this critical requirement.

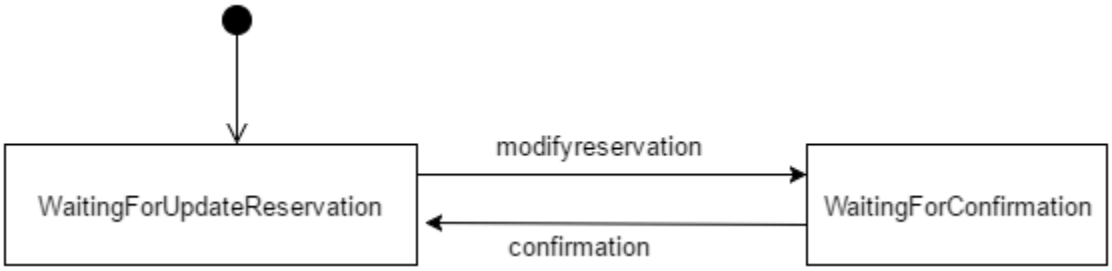
Use case UC3	Modify Reservation
Primary Actor	Student
Stakeholders and Interests	Student: Wants to modify reservation
Preconditions	- Student is identified and authenticated - Student has already made the reservation they desire to modify.
Success Guarantee (Postconditions)	
Main success scenario (or basic flow)	1 - Student select new timeslot from the reservation that needs to be modify. 2- System display confirmation message. 3- Student confirm 4- System display success message
Extensions (or alternative flows)	
Special Requirements	Touch screen UI
Technology and Data Variations List	Mobile app
Open issues	

UML system sequence diagram:

The actor first informs the system of the changes they require, only after they confirm it does the system apply the changes.



State Diagram:



System Operations:

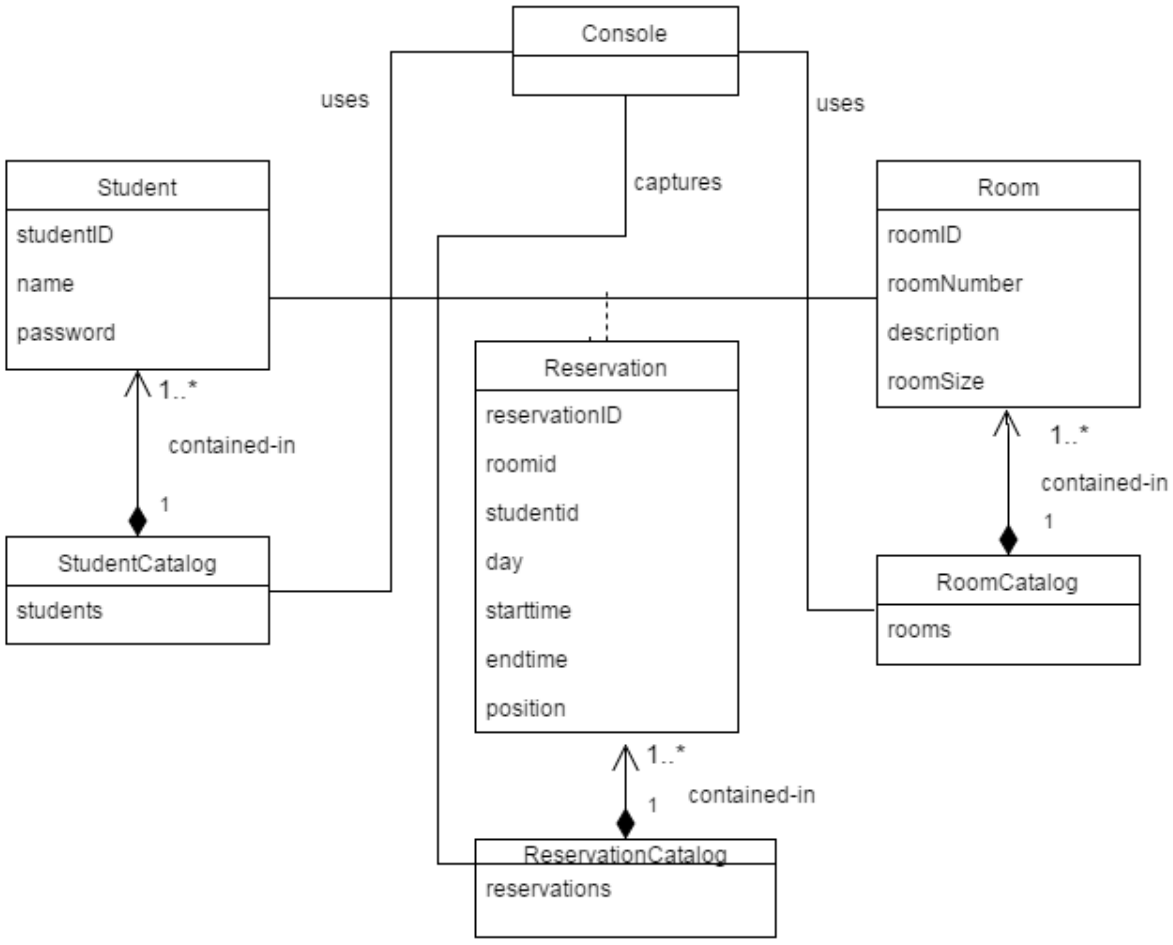
System Operations:
modifyReservation(studentId, oldReservationID ,newRoomId, newDay, newStartTime, newEndTime)

Operation Contract:

Contract	Modify Reservation
Operation	modifyReservation(studentId, oldReservationID ,newRoomId, newDay, newStartTime, newEndTime)
Cross References	UC3: Modify Reservation

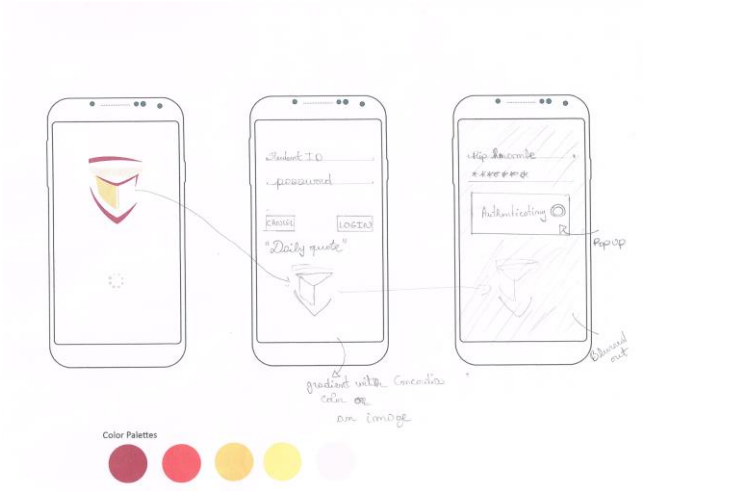
Preconditions	- Student has been authenticated - Student has chosen the reservation to modify - Student has chosen the new timeslot
Postconditions	- Reservation instance r was deleted. (instance deletion) - Attribute “position” of the reservations that has higher position number(lower in the waitlist) was decreased (Attribute modification) - A Reservation instance was created. (instance creation

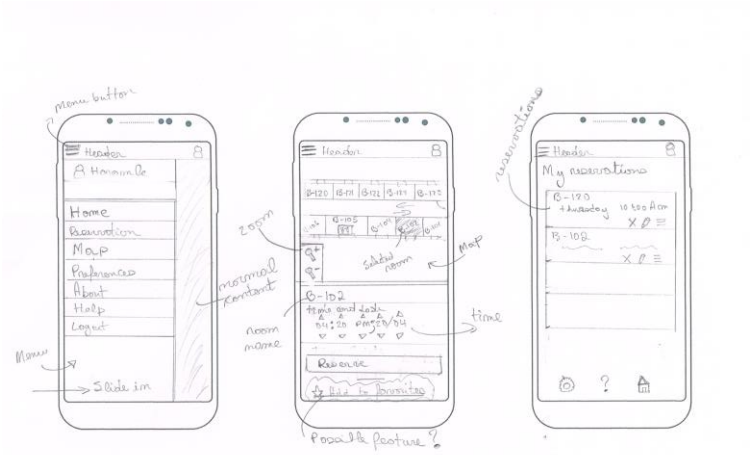
Domain Model

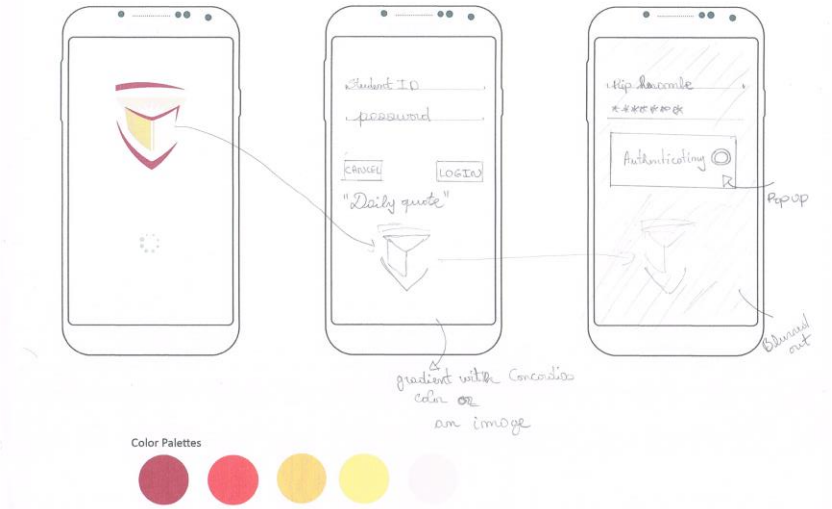


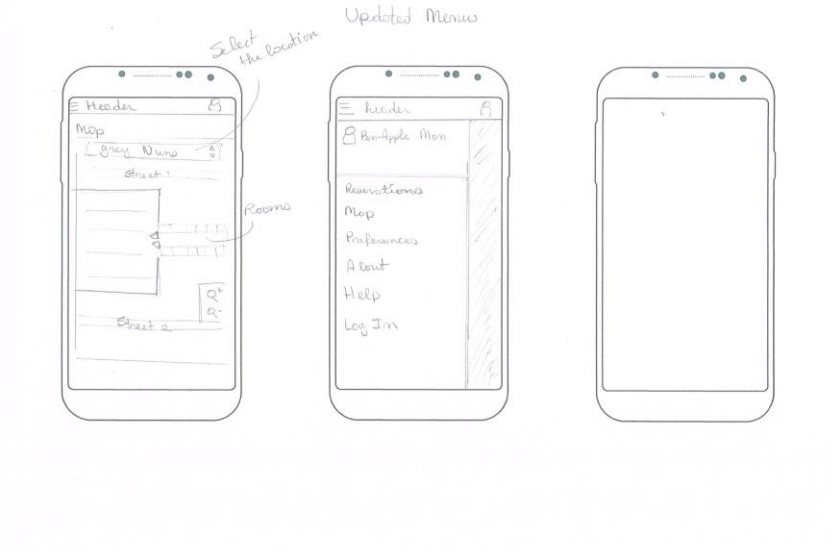
APPENDIX

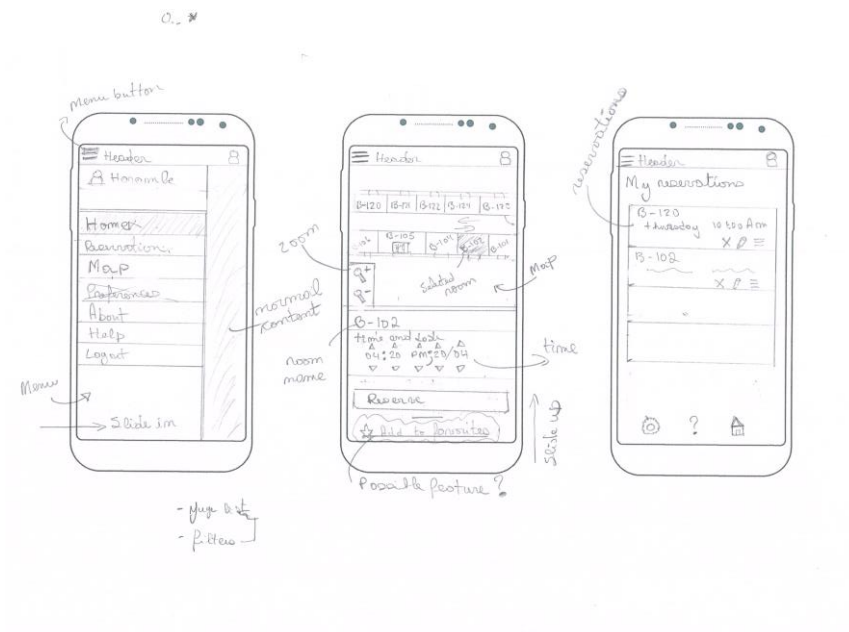
A- User Interface Mock-ups



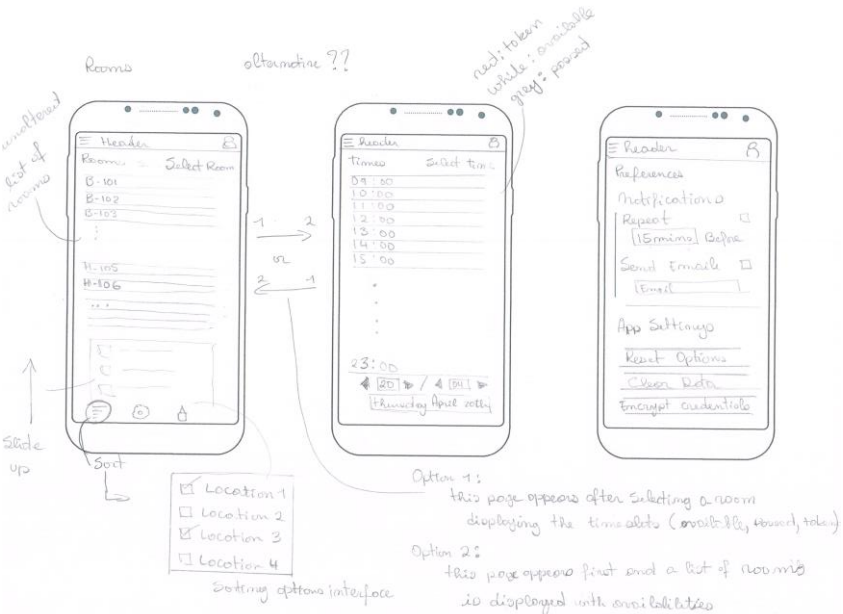



Student ID
password
CANCEL
LOGIN
"Daily quote"
Authenticating
Popup
gradient with Concordia
color or
an image
Color Palettes

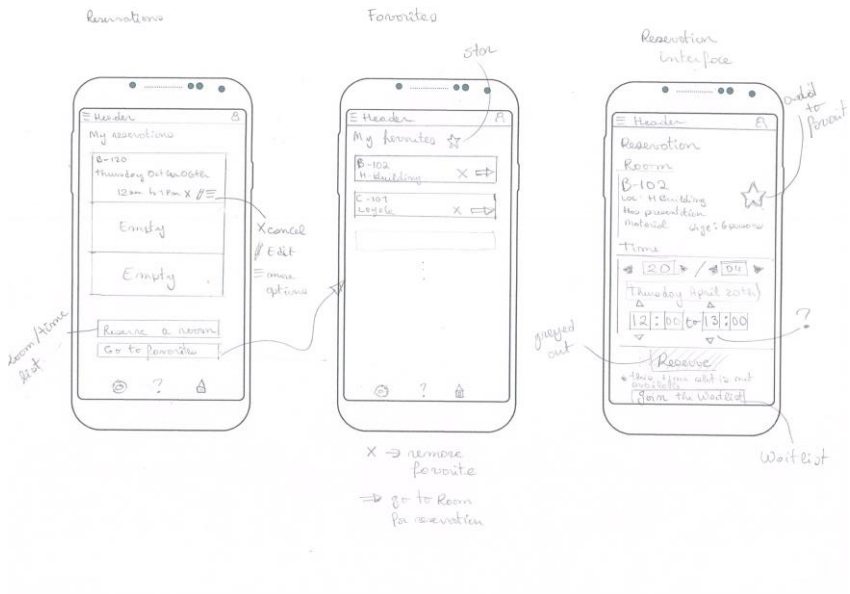

Updated Menus
Select the location
Header
Map
Grey Nuns
Rooms
Header
Reservations
Map
Preferences
About
Help
Log In



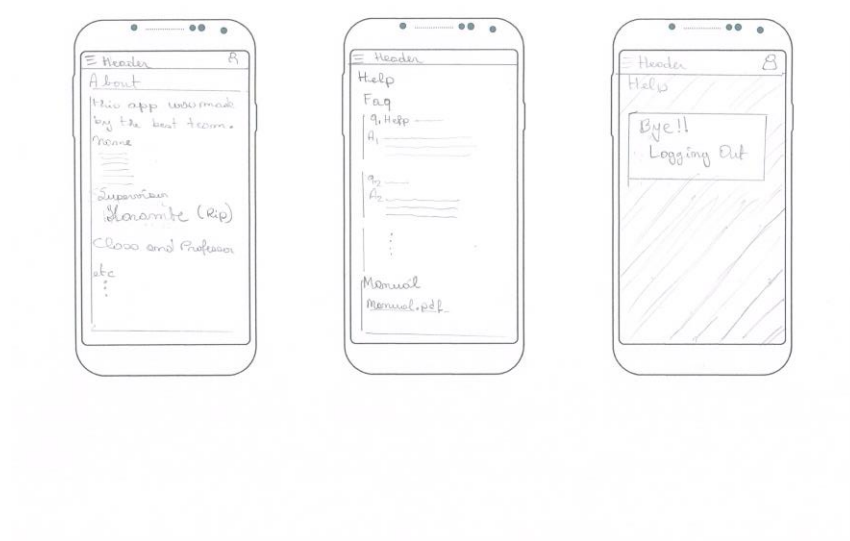
Menu button
Header
Hamamle
Home
Reservation
Map
Preferences
About
Help
Logout
Menu
Slide in
normal content
zoom
Selected room
Map
B-102
time and date
room name
time
Reserve
Add to favorites
Slide up
Possible feature ?
reservations
My reservations
B-120
B-102
- Yuge list
- filters



Rooms
alternative ??
unaltered list of rooms
Header
Room
Select Room
B-101
B-102
B-103
H-105
H-106
Slide up
Sort
Location 1
Location 2
Location 3
Location 4
Sorting options interface
Times
Select time
09:00
10:00
11:00
12:00
13:00
14:00
15:00
23:00
Thursday April 20th
Preferences
Notifications
Repeat
15mins Before
Send Email
Email
App Settings
Reset Options
Clear Data
Encrypt credentials
Option 1:
this page appears after selecting a room
displaying the time slots (available, passed, taken)
Option 2:
this page appears first and a list of rooms
is displayed with availabilities



Reservations
Favorites
Reservation interface
Header
My reservations
B-120
Empty
Empty
Reserve a room
Go to favorites
Xcancel
Edit
more options
Room/time list
Star
My favorites
B-102
H-Building
C-101
Loyola
X ⇒ remove favorite
⇒ go to Room for reservation
Reservation
Room
B-102
Time
20
04
Thursday April 20th
12:00 to 13:00
Reserve
Join the Waitlist
greyed out
Waitlist



Header
About
this app was made by the best team.
Name
Supervisor
Karamite (Rip)
Class and Professor
etc
Help
Faq
Manual
Manual.pdf
Bye!!
Logging Out