Software Testing Document (STD)

1. Introduction

In this document, we verify through testing, the different functionalities that were discussed in the SDD by referring to what the user is expected to input, and what the user should expect from the App to do. In addition to that, we assume several test cases (not just the ones typically inputted by the user) that would cover as much scenarios as possible (and thus verifying that the underlying events and conditions work as desired). This is also the best way to ensure robustness of our application in front of special situations such as when a user enters unexpected, invalid information, or when boundary values are involved. We are aware that our testing approach could mostly be considered a **black-box** one, rather than both white and black, but the adopted test sets guarantee a high percentage of condition coverage (from the high-level coding point of view); and this is related to the fact that the white-box approach requires a high level knowledge of internals of the software and under test and therefore certainly requires full access to the underlying code (this is something we lack in our case since the **lower-level code is hidden from developers working on the Bubble environment**).

2. Testing Procedure

Here we show all the test cases that were used per functionality (or module), while also displaying the generated result of the test from the User GUI's (whenever necessary). The requirements of each module is concisely mentioned right before applying the test cases (they can be found with greater detail in the SRS & SDD documents). A strong emphasis was given on proper **planning** of testing phase as entire success of final deployment depends on it.

In order to create real software development environment, different modules were assigned to different members. Each member was then assigned the responsibility of testing his own module. But for the sake of clarity, the final testing pictures in the following pages are being shown from the final application (as the unit testing was performed on very crude form of elements which were later integrated into the system as it appears now).

Signing Up

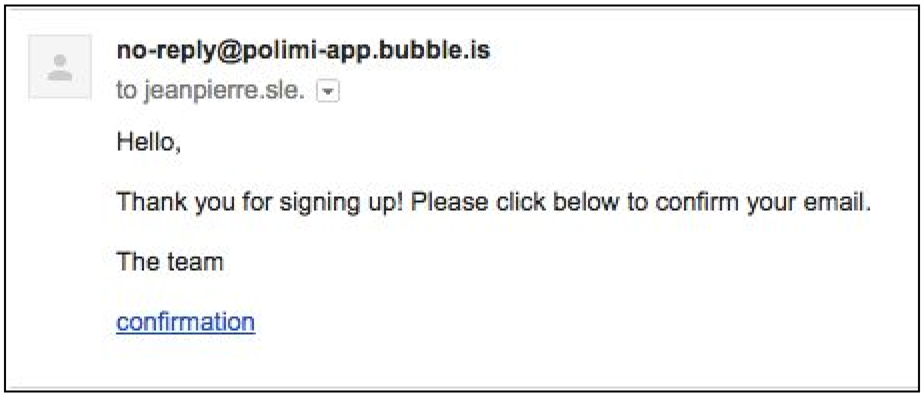
Requirements: The user enters a unique username, an email that can only be a Polimi email, and a password ⟶ The user is signed up and is sent a confirmation email

➢ **Test case 1**

<username = "jp", polimi email = "jeanpierre.sleiman@mail.polimi.it", password="123">

It is assumed there are no other registered users with either the same email or the same username.

SIGN UP Authentication: succeeds



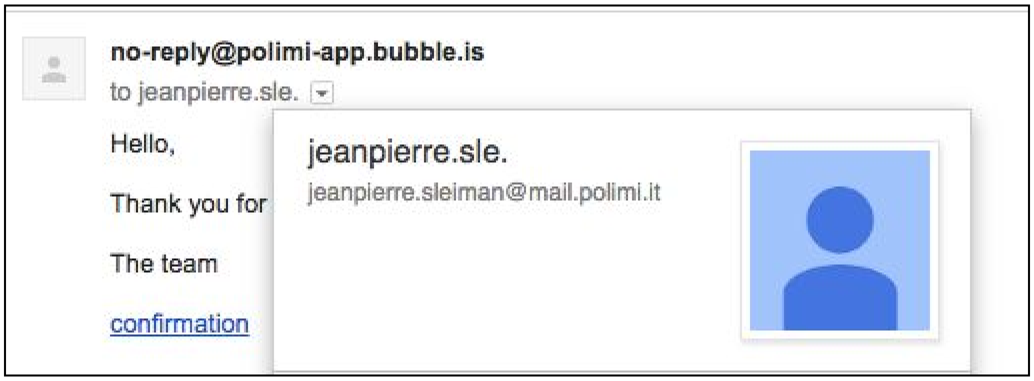
no-reply@polimi-app.bubble.is
to jeanpierre.sle.

Hello,

Thank you for signing up! Please click below to confirm your email.

The team

confirmation



no-reply@polimi-app.bubble.is
to jeanpierre.sle.

Hello,

Thank you for

The team

confirmation

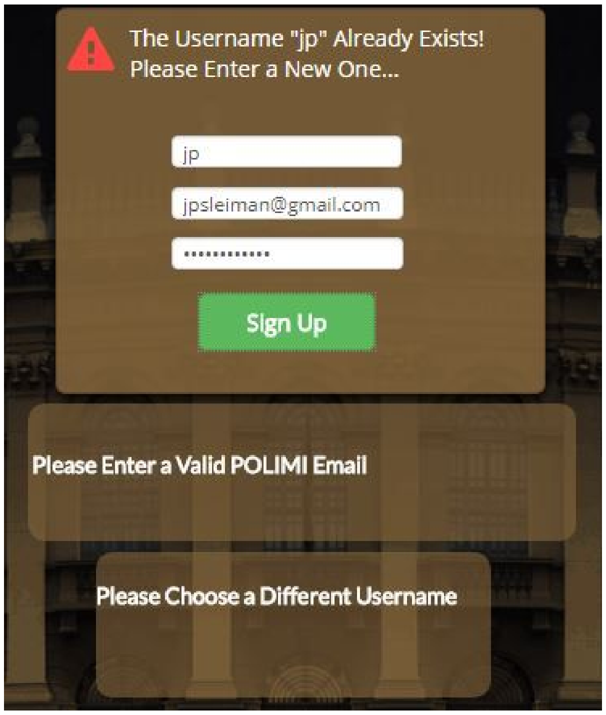
jeanpierre.sle.
jeanpierre.sleiman@mail.polimi.it

➢ **Test case 2**

<username = "jp", polimi email = "jpsleiman7@gmail.com", password="1234">

A user already exists with the same username.

SIGNUP FAILS *(invalid username and invalid email)*



The Username "jp" Already Exists!
Please Enter a New One...

jp

jpsleiman@gmail.com

············

Please Enter a Valid POLIMI Email

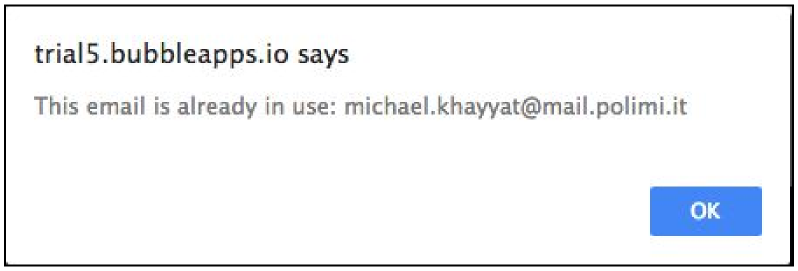
Please Choose a Different Username

➢ **Test case 3**

A user already exists with the same polimi email.

<username = "mkk", polimi email = "michael.khayyat@mail.polimi.it", password="1234">

SIGNUP FAILS *(email already in use)*



trial5.bubbleapps.io says

This email is already in use: michael.khayyat@mail.polimi.it

- ➢ **Test case 4**

<username = " ", polimi email = "jeanpierre.sleiman@mail.polimi.it", password="123">

Or if any of the other input components are empty.

SIGNUP FAILS *(empty component)*

Logging in

Requirements: The user enters his Polimi email with which he signed up, and enters the correct password as well $\longrightarrow$ The user is logged in and is taken from GUI 1 to GUI 2

- ➢ **Test case 1**

<polimi email = "jeanpierre.sleiman@mail.polimi.it", password="123">

It is assumed this email is already in the database (signed up) and the input password is the same as the one used during signup.

Note that the "Remember me" checkbox does not affect the Login functionality.

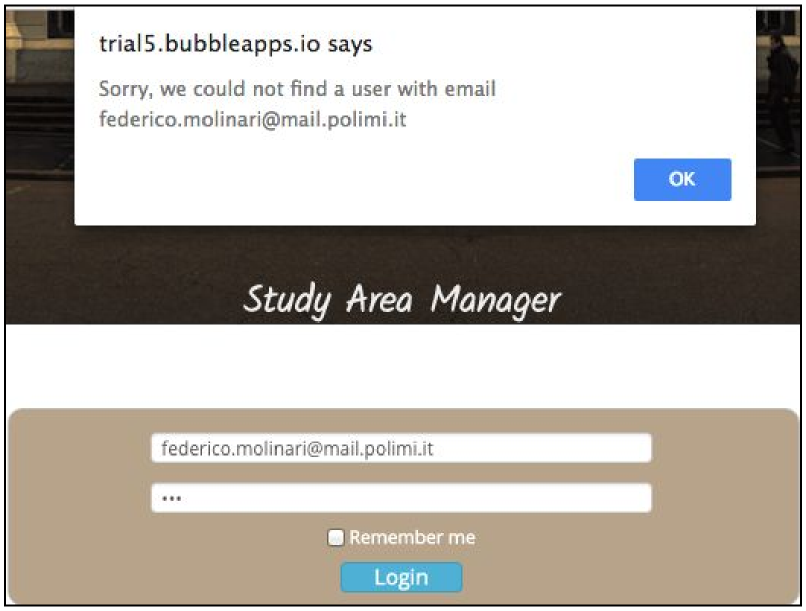
LOGIN SUCCEEDS

- ➢ **Test case 2**

<polimi email = "federico.molinari@mail.polimi.it", password="123">

Non-existing polimi email (in the database) is entered.

LOGIN FAILS *(email can't be found)*



trial5.bubbleapps.io says

Sorry, we could not find a user with email federico.molinari@mail.polimi.it

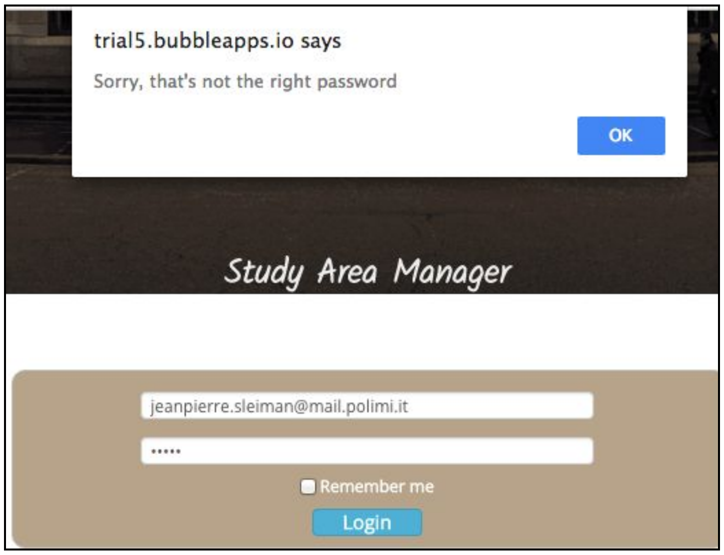
---

➢ **Test case 3**

<polimi email = "jeanpierre.sleiman@mail.polimi.it", password="hello">

Wrong password is entered.

LOGIN FAILS *(wrong password)*



trial5.bubbleapps.io says

Sorry, that's not the right password

Filtering

Requirements: The user inputs the desired search criteria in any combination possible in the dropdown menus and the input boxes ⟶ The study areas appearing on the page are filtered accordingly

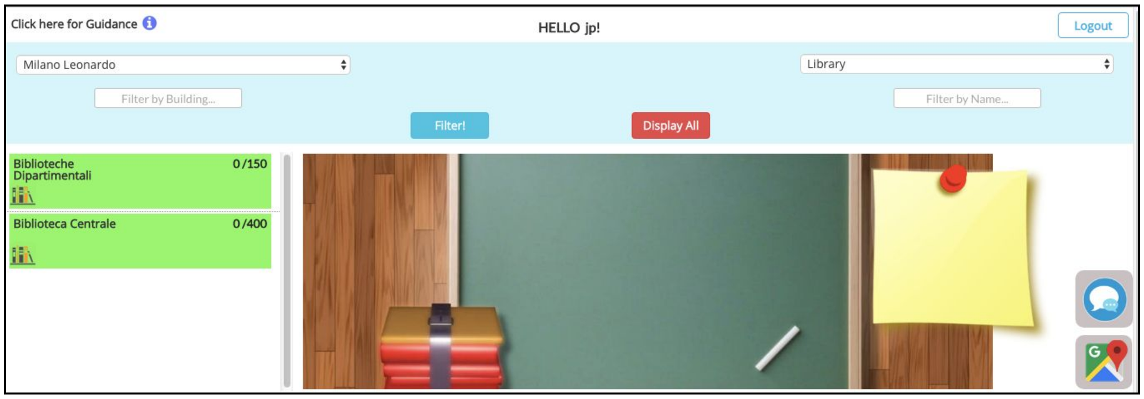
➢ **Test case 1**

<Campus = x, Category = y , Building = z, Name = w>

It is assumed in this test case that the different combination of choices among x, y, z, w (they can also be empty), yields a Study Area that can be found in the database.

For example:
<Campus = "Milano Leonardo", Category = "Library", Building = " ", Name = " ">

FILTERING SUCCEEDS *(the appropriate study-areas appear in the left column)*



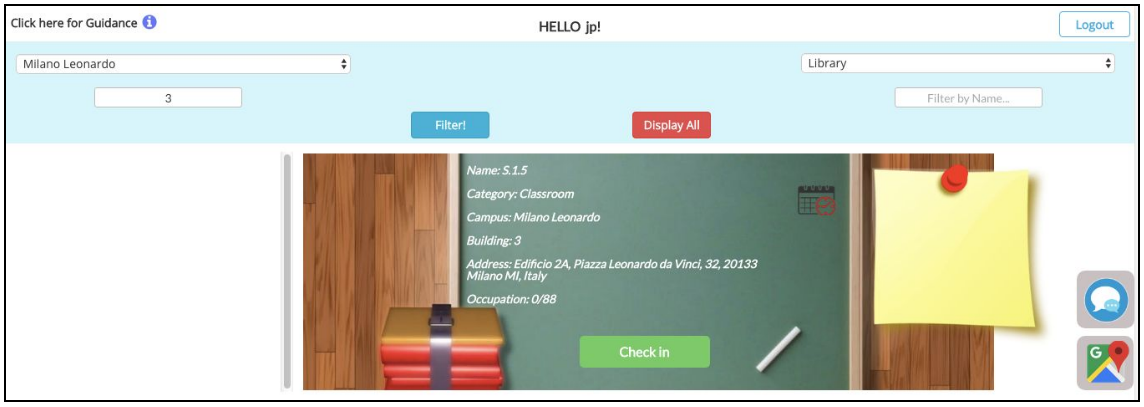
➢ **Test case 2**

<Campus = x, Category = y , Building = z, Name = w>

It is assumed in this test case that the different combination of choices among x, y, z, w yields a Study Area that cannot be found in the database.

For example:
<Campus = "Milano Leonardo", Category = "Library", Building = "3", Name = " ">
<Campus = " ", Category = " ", Building = " ", Name = "W.P.2">

FILTERING SUCCEEDS *(No Study Areas appear in the left column)*



Name: 5.1.5
Category: Classroom
Campus: Milano Leonardo
Building: 3
Address: Edificio 2A, Piazza Leonardo da Vinci, 32, 20133 Milano MI, Italy
Occupation: 0/88

Checking In/Out

Requirements:

Check-in: The user, not being already checked-in in any of the study areas, selects a desired option from the places listed on the page on condition that this place is available

⟶ The user is checked in to the selected place

Check-out: The user selects the study area in which he is already checked-in

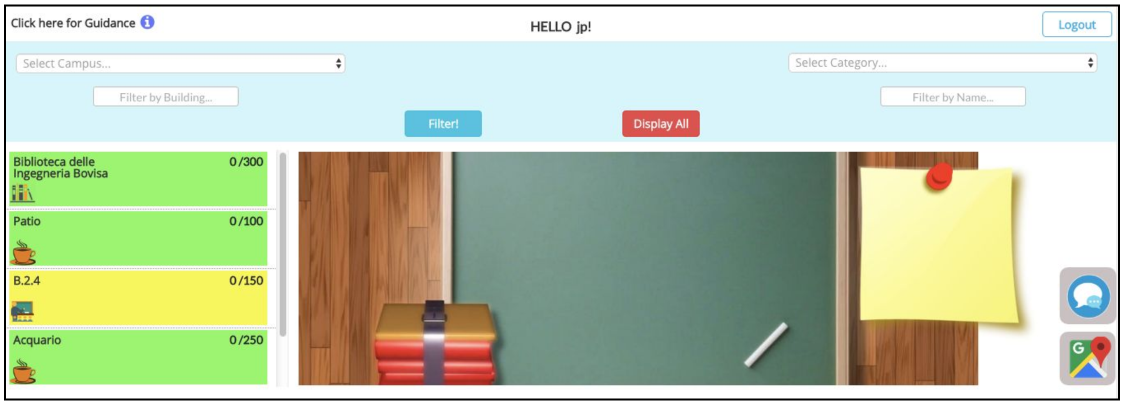
⟶ The user is checked out from that place

➢ **Test case 1**

<Selected Study Area = “ ”>

No study area has been selected yet.

CHECK IN not possible *(check in button doesn’t even appear yet)*



HELLO jp!

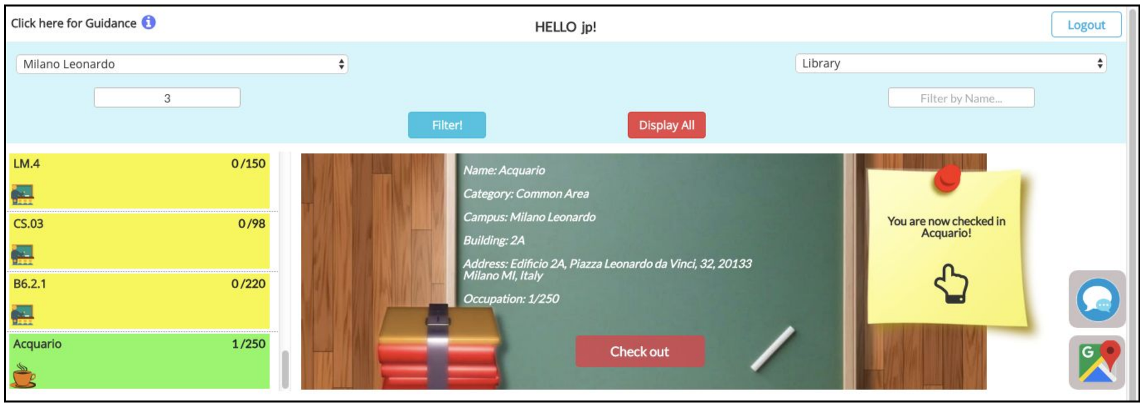
Biblioteca delle Ingegneria Bovisa	0/300
Patio	0/100
B.2.4	0/150
Acquario	0/250

➢ Test case 2

<Selected Study Area = "Acquario", Check in = true>

It is assumed that the user is not checked-in in any Study Area, and the number of occupants is 0 or any number less than the full capacity.

CHECK IN SUCCEEDS *(as described in the design document)*



HELLO jp!

Milano Leonardo

Library

3

LM.4	0/150
CS.03	0/98
B6.2.1	0/220
Acquario	1/250

Name: Acquario
Category: Common Area
Campus: Milano Leonardo
Building: 2A
Address: Edificio 2A, Piazza Leonardo da Vinci, 32, 20133 Milano MI, Italy
Occupation: 1/250

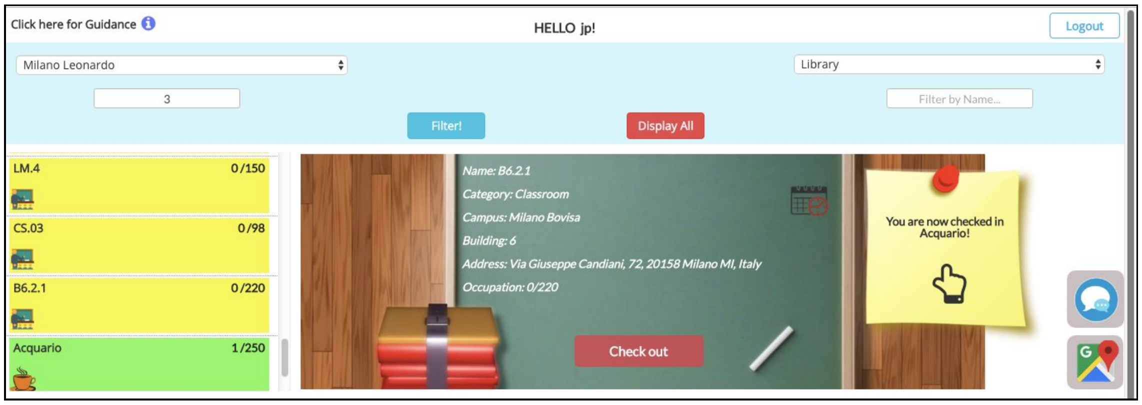
You are now checked in Acquario!

➢ Test case 3

<Selected Study Area = "B6.2.1">

The user is already checked in Acquario.

BROWSING OTHER STUDY AREAS while checked-in SUCCESS *(the student is still checked in Acquario but the information of B6.2.1 appears in the middle of the screen)*



LM.4 0/150
CS.03 0/98
B6.2.1 0/220
Acquario 1/250

Name: B6.2.1
Category: Classroom
Campus: Milano Bovisa
Building: 6
Address: Via Giuseppe Candiani, 72, 20158 Milano MI, Italy
Occupation: 0/220

You are now checked in Acquario!

➢ **Test case 4**

<Selected Study Area = “B6.2.1”, check out = true”>

The user is already checked in Acquario.

CHECK OUT FAILS *(no changes occur to the screen)*

➢ **Test case 5**

<Selected Study Area = “B6.2.1”, reload page = true>

The user reloads the page while already being checked in Acquario.

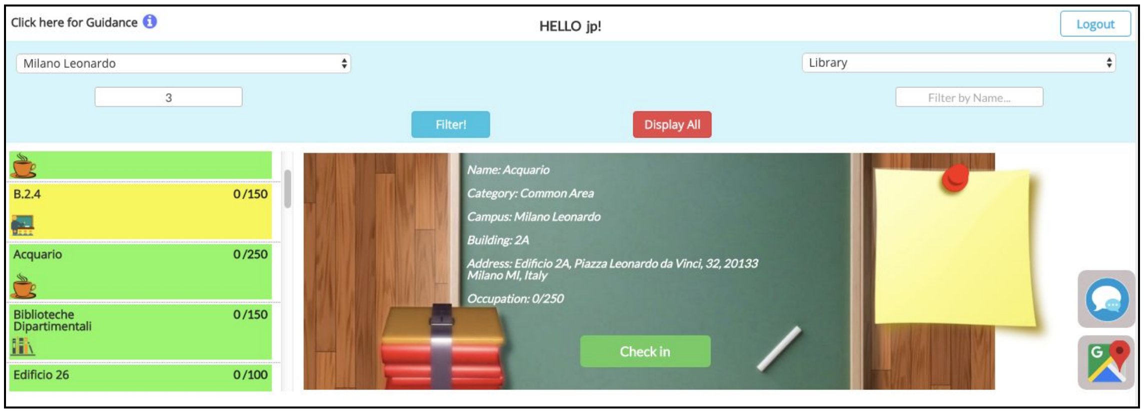
CHECKED IN STATUS IS PRESERVED *(the student is still checked in Acquario and its information appears in the middle of the screen)*

➢ **Test case 6**

<Selected Study Area = “Acquario”, check out = true”>

The user is already checked in Acquario.

CHECK OUT SUCCEEDS *(as described in the design document)*


Click here for Guidance

HELLO jp!

Logout

Milano Leonardo

Library

3

Filter by Name...

Filter!

Display All

B.2.4 0/150

Acquario 0/250

Biblioteche Dipartimentali 0/150

Edificio 26 0/100

Name: Acquario

Category: Common Area

Campus: Milano Leonardo

Building: 2A

Address: Edificio 2A, Piazza Leonardo da Vinci, 32, 20133 Milano MI, Italy

Occupation: 0/250

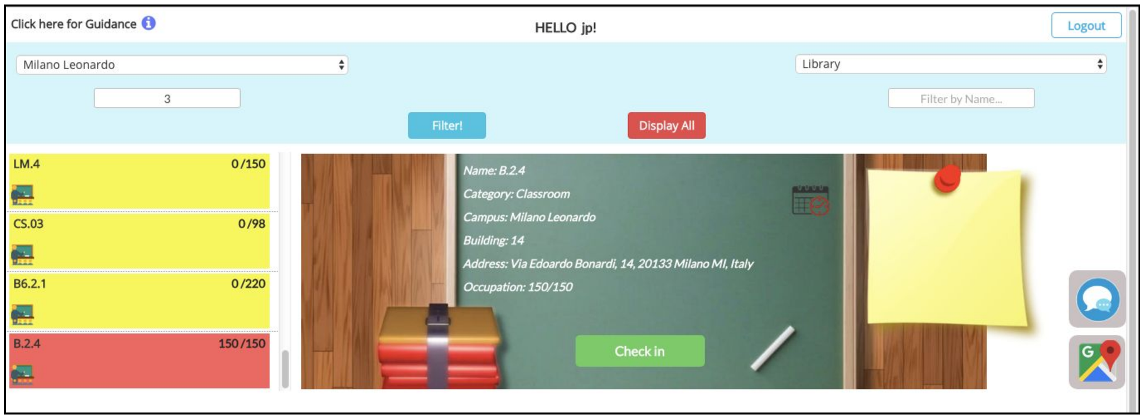
Check in

- ➢ **Test case 7**

<Selected Study Area = "B.2.4", check in = true">

It is assumed in this case that the study area has reached its full capacity (boundary value)

CHECK IN FAILS if study area is already full


Click here for Guidance

HELLO jp!

Logout

Milano Leonardo

Library

3

Filter by Name...

Filter!

Display All

LM.4 0/150

CS.03 0/98

B6.2.1 0/220

B.2.4 150/150

Name: B.2.4

Category: Classroom

Campus: Milano Leonardo

Building: 14

Address: Via Edoardo Bonardi, 14, 20133 Milano MI, Italy

Occupation: 150/150

Check in

Searching by Distance

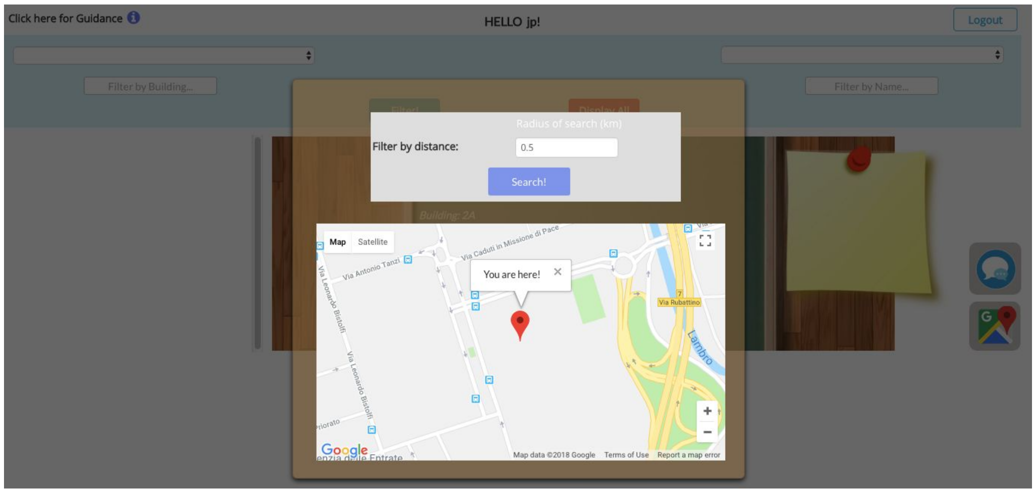
Requirements: The user enters a valid radius of search (greater than zero) ⟶ *The study areas are filtered accordingly and are shown on the map*

- ➢ **Test case 1**

<Radius of search = x>

It is assumed that x is not big enough to encompass any of the listed study areas in the database.

SEARCH SUCCEEDS *(nothing appears on the map except for the user's current location, and nothing appears in the left column of the GUI)*



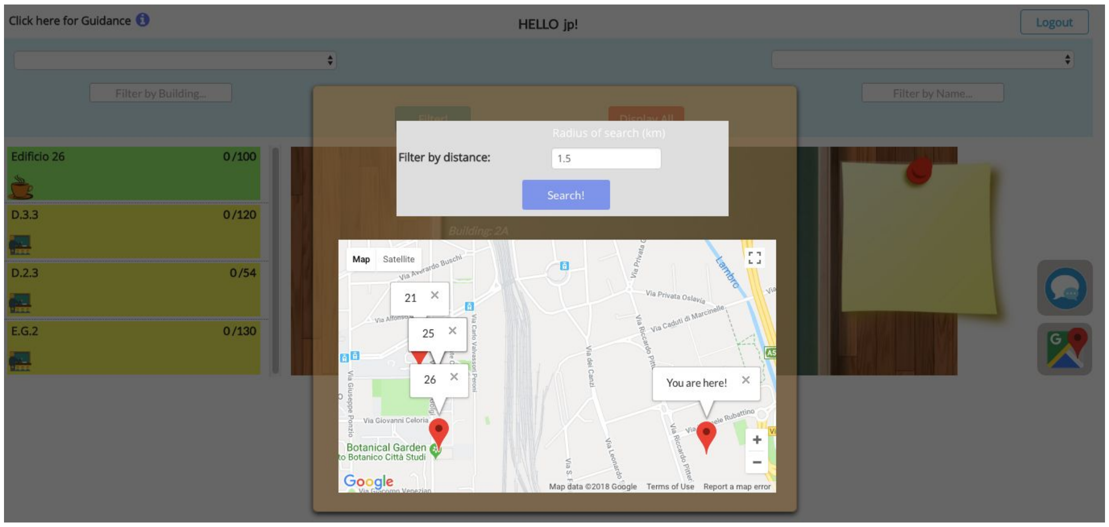
Filter by distance: 0.5

➢ **Test case 2**

<Radius of search = x>

It is assumed that x is big enough to encompass some of the listed study areas in database.

SEARCH SUCCEEDS



Filter by distance: 1.5

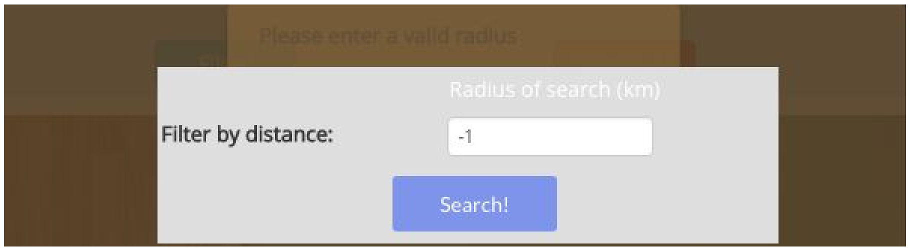
Edificio 26	0/100
D.3.3	0/120
D.2.3	0/54
E.G.2	0/130

➢ **Test case 3**

<Radius of search = x>

It is assumed that x is invalid (i.e. less than or equal to zero)

SEARCH FAILS *(an error message appears)*



Posting/Deleting Messages

Requirements:

Posting: The user selects any desired study area from the page, and types a certain message ⟶ The message is posted and appears for all users who select the same study area

Deleting: The user goes to any message that he has himself posted previously ⟶ *The message is deleted and thus disappears from the post box for all users*

➢ **Test case 1**

<message = "Hello", student = "jp", selected post place = " ">

No study area has been selected yet.

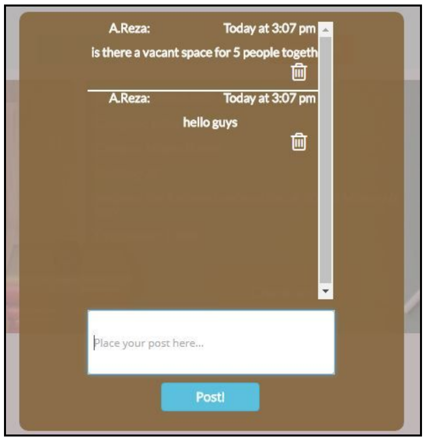
POSTING FAILS *(no option for posting messages appears)*

➢ **Test case 2**

<message = “I forgot my phone in Patio, if anybody finds it please let me know?”, student = “jp”, selected post place = “Patio”, post = true>

It is assumed the current user is “jp”.

POSTING SUCCEEDS



A.Reza: Today at 3:07 pm
is there a vacant space for 5 people togeth

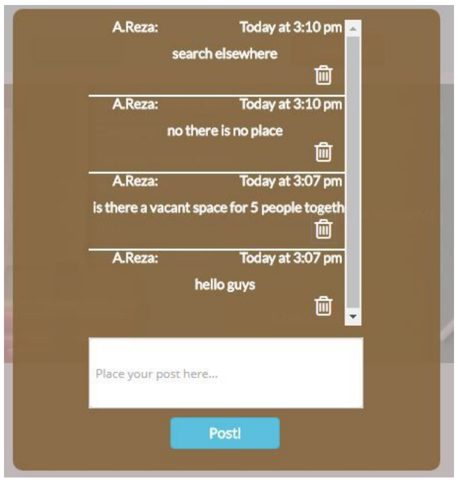
A.Reza: Today at 3:07 pm
hello guys

➢ **Test case 3**

<message = “I forgot my phone in Patio, if anybody finds it please let me know?”, student = “jp”, selected post place = “Patio”, delete = true>

It is assumed the current user is “jp” ofcourse.

DELETING SUCCEEDS

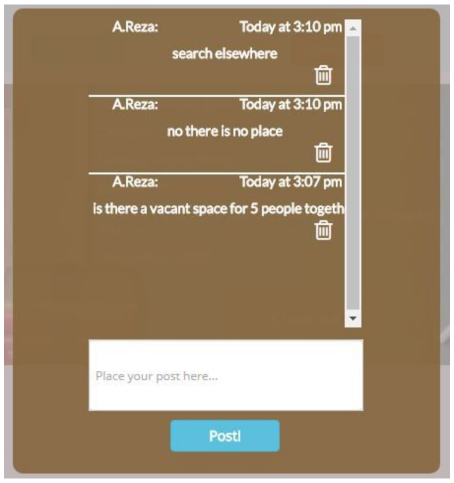


A.Reza: Today at 3:10 pm
search elsewhere

A.Reza: Today at 3:10 pm
no there is no place

A.Reza: Today at 3:07 pm
is there a vacant space for 5 people togeth

A.Reza: Today at 3:07 pm
hello guys



A.Reza: Today at 3:10 pm
search elsewhere

A.Reza: Today at 3:10 pm
no there is no place

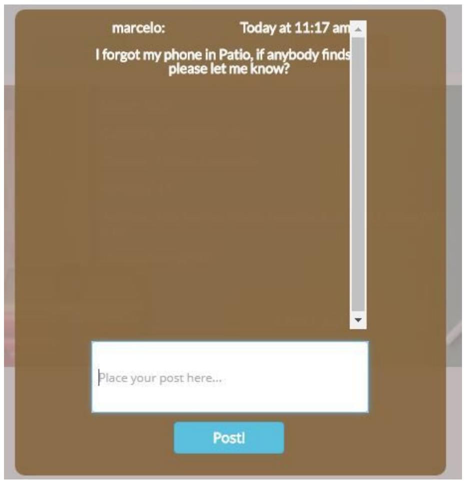
A.Reza: Today at 3:07 pm
is there a vacant space for 5 people togeth

➢ **Test case 4**

<message = "I forgot my phone in Patio, if anybody finds it please let me know?", student = "marcelo", selected post place = "Patio" >

It is assumed that the current user is "jp".

DELETING FAILS *(the delete icon doesn't even appear since students are not allowed to delete other user's messages)*


marcelo: Today at 11:17 am

I forgot my phone in Patio, if anybody finds please let me know?

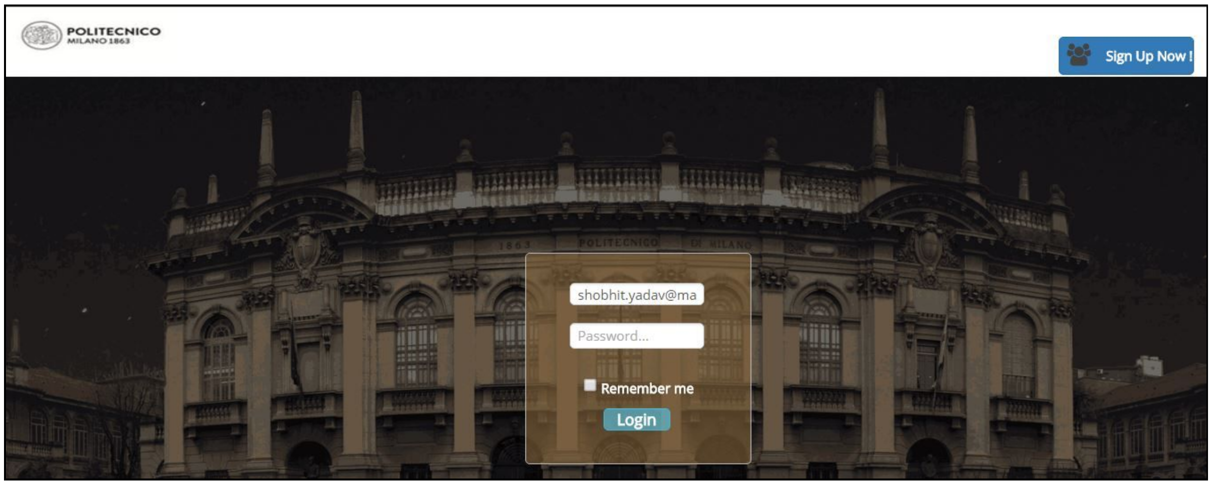
----

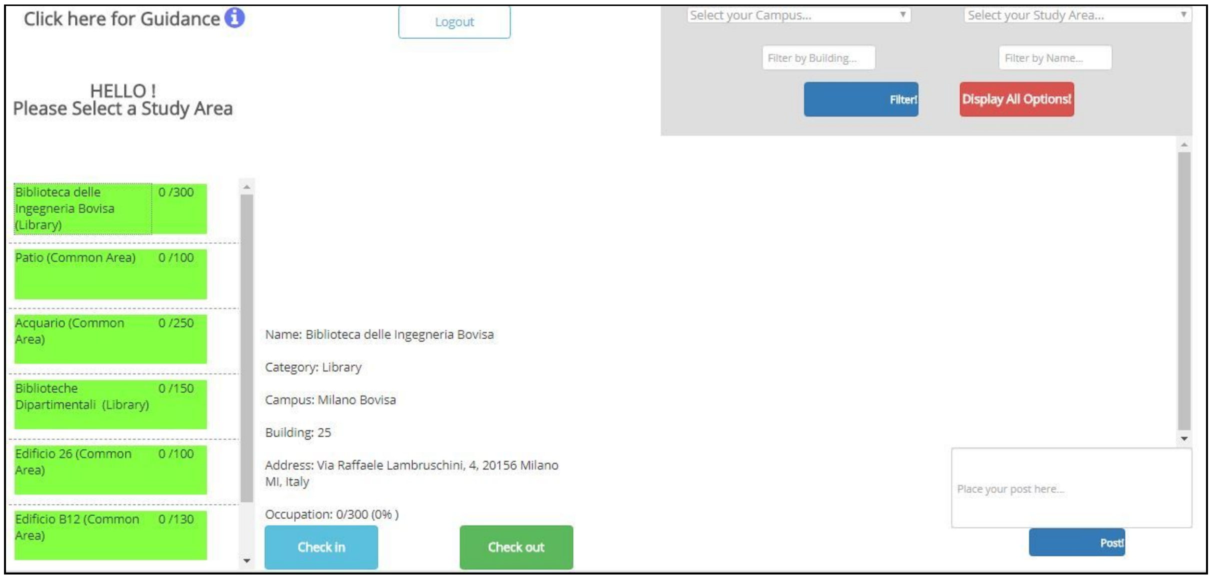
3. Experimental Trials

Before concluding this document, and therefore all our project work, it is worth mentioning that during the design phase and after the design completion, we carried out certain tests with software engineering professional and Polimi Students (our end users). In both the test, the idea was to obtain necessary feedbacks. It was quite satisfactory for us to see how positively they reacted, and how helpful they were in their constructive feedback. In fact, unanimously the people who tried our application found the idea (the project concept) itself very attractive as they could relate with us in terms of the motivation behind developing this interface.

Alpha-test

In this section we present one of the first drafts of our application, it contains most of the functions that were required according to the specifications and then was forwarded to our colleagues who work as software engineers. This helped us to get a professional advice and a simulation of a real alpha-test. An image of the **alpha-version** of our application is given below. As can compared from the interface used in prior test, alpha version was evolved significantly and thus improved a lot.





HELLO !
Please Select a Study Area

Biblioteca delle Ingegneria Bovisa (Library)	0 /300
Patio (Common Area)	0 /100
Acquario (Common Area)	0 /250
Biblioteche Dipartimentali (Library)	0 /150
Edificio 26 (Common Area)	0 /100
Edificio B12 (Common Area)	0 /130

Name: Biblioteca delle Ingegneria Bovisa

Category: Library

Campus: Milano Bovisa

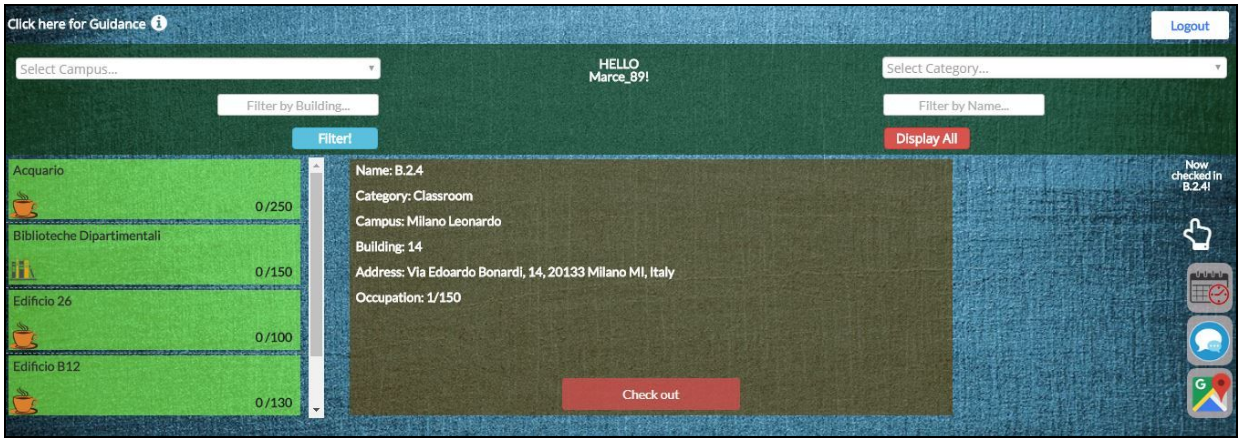
Building: 25

Address: Via Raffaele Lambruschini, 4, 20156 Milano MI, Italy

Occupation: 0/300 (0%)

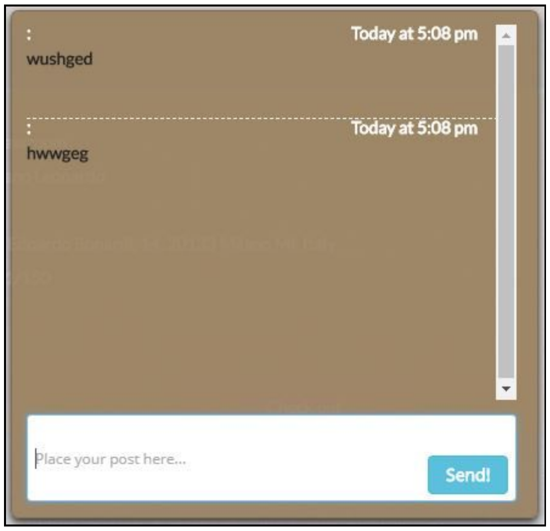
Beta-Test

Here we tested our application with the real customers i.e. the Polimi Students which are the end users of PoliSAM. they found the **GUI to be easy and intuitive** to understand and operate with; also none of them complained about any problems related to the different modules performing their required **functionalities according to the defined specifications**. However, most of them thought that the graphical interface could be visually ameliorated (in terms of the front-end graphics); and so this is why we moved from the GUI 2 that was presented in the SDD document, to a more improved version which was displayed in the Unit testing section of this document.

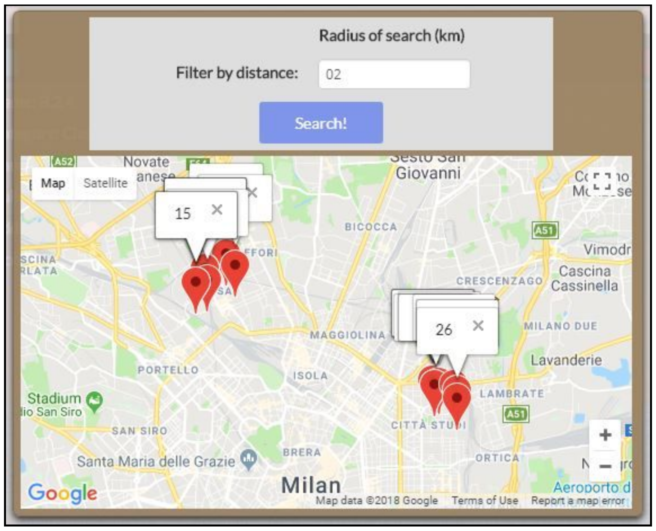


Acquario 0/250
Biblioteche Dipartimentali 0/150
Edificio 26 0/100
Edificio B12 0/130

Name: B.2.4
Category: Classroom
Campus: Milano Leonardo
Building: 14
Address: Via Edoardo Bonardi, 14, 20133 Milano MI, Italy
Occupation: 1/150


: Today at 5:08 pm
wushged

: Today at 5:08 pm
hwwgeg



---

4. Conclusion

Effort Distribution in different phases

As mentioned during the course, we could totally relate with the effort distribution during the software development phase. After working on a real software project, we realized experientially that the perceived general view that coding is the most important phase for a software development is incorrect. In fact, a significant amount of time was devoted for setting up the requirements specification and testing.

Deployment

Our responsibility regarding this phase of software lifecycle, we do not have enough workload as it can be for softwares that need installation or developed for special customers. Our application can be simply accessed through Polimi Online Services and through all possible electronic devices.

Maintenance

One of the most important features of the PoliSAM is easy maintenance. Also fixing bugs can be easy and fast, owing to the chosen visual programming platform. Also no adept personnel are required if a bug is identified and the developers being students are not available. Hence, maintenance is not cost intensive. Although, there is always a scope of corrective maintenance generating from evolution of environment. Before the final deployment planning for maintenance should be formulated.

Outcome

The most important outcome for us was **learning the software development process** being new to the field. On the other hand, the application successfully **delivers the specified requirements**. It is scalable, portable, ergonomic and modular that ensures incrementality. By the conducted tests we also ensured the **robustness** of the application for any possible interaction of the user. The **reliability** of the application is highly **dependent on the assumption that students act correctly.**

In addition, we will like to highlight the possible problems we can face due to the chosen working platform i.e. **Bubble**. We can expect to have some performance **limitations** that can affect scalability. Some **specific issues from design phase** are follows:

- We desired to use "*username*" for login but the working platform limits us to use the email id for that purpose.
- We also faced limitations in front-end design and consequently we had to avoid usage of some elements. This was particularly felt during the responsive design of the application.
- Also the final rendering of the application in live version was not really intuitive as sometimes results achieved were deviating from the expected ones (in context of front-end design). This was particularly felt to make application work on mobile.

We were although able to solve the above mentioned problems for the task at hand but in general no strong assertions can be made for the success of the platform. On the other hand, traditional programming languages normally are flexible enough to deal with these situations.