SMART Co-working Space

**Table of Contents**

[Chapter 1: Introduction 4](#_Toc48023280)

[1.1 Background and Motivation 4](#_Toc48023281)

[1.2 Problem Statement 4](#_Toc48023282)

[1.3 Scope 5](#_Toc48023283)

[1.4 Project Objectives 5](#_Toc48023284)

[1.5 Methodology 6](#_Toc48023285)

[1.6 Solution 7](#_Toc48023286)

[Chapter 2: Problem Analysis 9](#_Toc48023287)

[2.1 Problem stakeholders 9](#_Toc48023288)

[2.2 Main User Groups 9](#_Toc48023289)

[2.3 Requirements Gathering 10](#_Toc48023290)

[2.3.1 Interview 10](#_Toc48023291)

[2.3.2 Online Survey 10](#_Toc48023292)

[2.3.3 Observation 10](#_Toc48023293)

[2.3.4 Ethnography 11](#_Toc48023294)

[2.5 Functional Requirements 11](#_Toc48023295)

[2.5.1 End User: 11](#_Toc48023296)

[2.5.2 Co-working Space’s Owner: 12](#_Toc48023297)

[2.5.3 System Functions 13](#_Toc48023298)

[2.5.4 General Functions 14](#_Toc48023299)

[2.6 Non-Functional Requirements 15](#_Toc48023300)

[Chapter 3: Requirements Modeling 16](#_Toc48023301)

[3.1 System Architecture Diagrams 16](#_Toc48023302)

[3.2 Use cases Diagrams 17](#_Toc48023303)

[3.3 Use case Tables 18](#_Toc48023304)

[3.3.1 Admin 18](#_Toc48023305)

[3.3.2 Manager 29](#_Toc48023306)

[3.3.3 User 30](#_Toc48023307)

[3.4 System Context Diagram 38](#_Toc48023308)

[3.5 Sequence Diagram 39](#_Toc48023309)

[3.5.1 Sign up 39](#_Toc48023310)

[3.5.2 Login 41](#_Toc48023311)

[3.5.3 Booking 42](#_Toc48023312)

[3.5.4 Booking cancellation 43](#_Toc48023313)

[3.5.5 Add co-working space 44](#_Toc48023314)

[3.5.6 Search for place 45](#_Toc48023315)

[3.5.7 Update profile 46](#_Toc48023316)

[3.5.8 Add review /comment 47](#_Toc48023317)

[3.6 Class Diagram 48](#_Toc48023318)

[3.7 Entity relationship Diagram 49](#_Toc48023319)

[Chapter 4: Design and Issues: 50](#_Toc48023320)

[4.1 Covering tools / platforms used: 50](#_Toc48023321)

[4.2 Issues (Challenges) 50](#_Toc48023325)

[Chapter 5: System Testing and Evaluation: 51](#_Toc48023333)

[5.1 Testing: 51](#_Toc48023334)

[5.2 Planning 52](#_Toc48023335)

[5.3 Lesson Learnt 53](#_Toc48023336)

[5.3.1 Technical Skills: 53](#_Toc48023337)

[5.3.2 Non-Technical Skills: 53](#_Toc48023338)

[5.4 Future work 54](#_Toc48023339)

# Chapter 1: Introduction

## Background and Motivation

*What is a Co-working Space?*

*In Google’s dictionary terms co-working is a type of work that includes “the use of an office or other working environment by people who are self-employed or working for different employers, typically so as to share equipment, ideas, and knowledge”.*

*-Google.*

Co-working Space provides individuals or groups with desk space, meeting rooms, shared/private spaces and offices. Allowing them to have a convenient, cost efficient and flexible working environment. It is like a simulation of a real-world office, where you find people from different ages and categories, come to accomplish their needs; students come to study, share ideas or brainstorm, companies/start-ups/student activities make interviews, meetings or give sessions, and so on.

Now, after we have introduced the idea of Co-working spaces, lets discuss how it is commonly used and became very popular nowadays. According to Wikipedia: *"Between 2006 and 2015, a few studies have shown the number of co-working spaces and available seats have roughly doubled each year"*, proving the popularity and demand they gained over the past few years.

With this evolution of such places, it has to exist a system to organize the process internally and externally. Internally where the owner of the space can manage and control the transactions made within the system (CRUD operations, view charts and tables, monitor profit). Externally where we facilitate the process for the end user, when he/she desire to make/cancel a reservation, make comparisons with other spaces, browse, live preview of the system and so on. Thus, we thought to implement a system represented by our Website and Mobile App to help both the owner of the place and the end user, by making the process easy, user-friendly and fun to use.

## Problem Statement

Recently, the need for such places for meetings, studying, training courses, interviews, etc. has increased significantly, as we have discussed earlier. As a result, these places were noticeably so crowded.

**From The End User Side:**

People have to follow traditional ways of communication, whether through the phone or through the use of social media platforms or by the primitive method where they go by themselves to that place inquiring about the place features and services such as: available rooms, prices, capacity, existence of internet connection, white boards, white boards markers and so on. Also, if the user is confused and doesn’t know which place will be suitable for his needs, following the above ways to decide will increase his confusion, beside wasting of time and effort and lack of efficiency.

Furthermore, following these ways prevent or reduce the connection between place and end user, when an emergency situation happens, such as: internet, electricity or water dis-connectivity; which may cause loss of customers, consequently decrease in profits.

**From The Place Owner Side:**

They suffer from the lack of a regulatory system for booking, especially when there are more than one principal. So, it leads to randomness and chaos. In addition to following same traditional ways of communication with clients. Besides, still depending on paper work, which is a bad and outdated approach and also lack of a system which provides him with graphs or charts to evaluate and monitor place performance.

## Scope

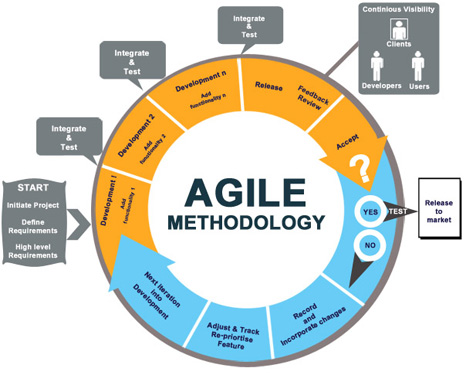
* Developing a web application for both end user and owner.
* Developing a mobile application for end user.
* We are targeting people that most interested in co-working spaces like students, startups, student activities groups.

## Project Objectives

* Enabling the user to be open to many coworking spaces as he wishes, from only one place (our system).
* Regarding Usability Goals:
* ”is the ease of use and learnability of a human-made object such as a tool or device”.
  + Efficient and Effective to use.
  + Safe to learn.
  + Easy to learn.
  + Easy to remember how to use.
  + User friendly interface.
* Regarding User Experience goals (UX):
* ” The way people feel about it and their pleasure and satisfaction when using it”. The user will find the system:
  + Fun.
  + Satisfying.
  + Enjoyable.
* Regarding Functionality goals:
  + Facilitate the reservation process.
  + Provide real-time coverage of the condition of the place.
  + Facilitate the search process for places.
  + Recommend similar places to user based on certain criteria.
  + Provide administrative features for the place owner in order to evaluate place’s performance.
* Meeting users’ mental models.
* Regarding Critical Success Factors (CSF) :
  + 80% of site visitors complete a reservation.
  + 90% Ensure happy and satisfied users.
  + 50% Recommend the app/website to a friend.
  + 60% Use it again and not go to other competitors.
* Say goodbye to routine paperwork.

## Methodology

We use agile model as our life cycle ,because **Agile software development** describes an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing cross-functional teams and their customer(s)/end users(s).It advocates adaptive planning, evolutionary development early delivery, and continuous improvement, and it encourages rapid and flexible response to change, This include the following sprints:

**Project Introduction**

* Identify the problem
* Identify the objective
* Identify project scope
* Stakeholders
* Project planning

**Project Analysis**

* Identify functional requirements
* Identify non-functional requirements
* Identify use case diagram
* Identify class diagram
* Identify sequence diagram

**Development**

We Divide our project into sprints in each sprint we

* Clarify the detailed requirements for each sprint
* Analysis
* Design
* Implementation
* Testing
* Integration

## Solution

Considering the given inputs and real-life problems we have discussed earlier in problem statement, we thought it is better to implement a system which helps both the owner of the place and also the end user visiting these places.

This system provides us with an online reservation system, enabling the user to easily do the basic operations of reservation process, without the obligation of searching Google or Facebook to browse or communicate with these places. In our case, we will make these operations as easy as possible and at one place; the user may search, browse, compare, reserve, add review, communicate with the place through our website. In other words, we will enable the user to be open to many co-working spaces as he wishes, from only one place, so guaranteeing fun and easy UX and fatigue free.

On the other side, We didn’t forget the owner of the place, and how much his mission is hard and important ; So we decided to engage some features that mean to him and will ease his work; as providing him with a dashboard to control and monitor the work flow, also showing graphs and charts representing some aspects as performance, traffic, sales, websites’ visitors and so on.

# Chapter 2: Problem Analysis

## Problem stakeholders

* The Co-working space communities that are providing the services to the user in actions, that will interact with our system to register a new co-work space.
* Users: Normal users that will interact with the system to reserve a place in the chosen co-work space and many other functionalities, they can be categorized as students, student activity groups, start-ups companies doing some interviews, normal users doing their stuff and so on…

## Main User Groups

Customer

Co-working Space Admin

Co-working Space Manager

|  |  |  |  |
| --- | --- | --- | --- |
| Characteristics | Customer | Co-working Space Admin | Co-working Space Manager |
| Age / gender | 50%male- 50% female  18-25 | 75% male-25% female  20-30 | 50% male-50% female  20-30 |
| Education | Med | Med | Med |
| language | Arabic | Arabic | Arabic |
| Computer / web experience | Med | Med | Med |
| Task knowledge | First week: Low to Med then Med to High | First week: Low to Med then Med to High | First week: Low to Med then Med to High |
| Expectation | Ease of use | Ease of use | Ease of use |

## Requirements Gathering

### Interview

To get the real requirements that the actual user needs for the system and in our case is the owner of the co-working space, office Admin.

1. What problems do you face every day in managing the place?
2. What do you think is the reason for these problems?
3. How bookings are recorded?
4. What factors do you consider for making a particular offer?
5. What if there is a system to help you to manage the place and book reservations easily?
6. Now I can show him the proposed features of that system to make sure its priority for him.

### Online Survey

To get all views of the largest number of users in less time.

1. Do you use the co working spaces?
   1. A lot
   2. Little
2. If there is a reservation system for those places that make it easier for you to book, what would you prefer?
   1. Website
   2. Mobile APP
3. Why did you choose that choice?
4. Do you want to a particular addition in this system?

### Observation

- To understand the task that person is doing in the place and the problems he faces every day.

- An Example of observation guide framework:

Who: Office Admin?

Where: Co working space?

What: The process of receiving and recording reservations?

- What is his role?

- How to receive bookings?

- What is used to record this data?

- What is the complexity of this process?

### Ethnography

- To be able to understand system users deeply as the experiment is the best thing to learn.

- It is possible to have this experience by working a team member for a day or more in those places to understand the life cycle of booking process, the user's needs and problems they are face every day.

## Functional Requirements

### End User:

* Make Reservation
  + User opens desirable co-working space and selects the best fit asset for him\here like meeting room, private office etc.
  + User enters booking data such as duration, starting time, end time, number of people and confirm booking.
* Cancel Reservation
  + User can cancel or delete his reservation after some period of time defined by co-working owner.
* Make Payment
  + User is allowed to pay online or cash, in case of credit card user enters his\her payment information.
* Review Co-working Space
  + User is able to review or rate co-working spaces.
* Search for Co-working Space
  + User is able to search for co-working spaces by name, address.
* View available offers
  + User can view available offers for co-working spaces.
* View Co-working Space on Map
  + User can view the selected co-working space on map.
* Add Co-working space
  + User can add his own co-working space by providing the required co-working data like location, name, features and assets etc.
* Update Profile Settings
  + User can update his profile settings like changing name, address, email, password or any other attributes.
* View Booking History
  + User can view his\her booking history over time across the system.
* Customer Support
  + User can send complain issue to specific co-working space or about the system in general.

### Co-working Space’s Owner:

* Verify Reservation
  + Admin can verify reservation in case of cash payment by user.
* Delete Reservation
  + Admin can delete some reservations.
* Retrieve Reservations
  + Admin can retrieve new, history, canceled reservations for further operations.
* Manage forecasting next month requests
  + A data mining model that helps owner forecasting next month booking request by selecting training data range and resource.
* View monthly profit
  + Admin can watch profit and number of booking requests tracking divided monthly.
* Send Email to User
  + Admin can send email to end user about his\her reservation for further details.
* Add New Current View
  + Admin can add a view of his\here co-working space at the current state through images.
* Manage Co-working Desks
  + Add: Admin can add a new seating to his\here co-working space like meeting or private by entering capacity, price, type etc.
  + Retrieve, Update, delete seating.
* Manage Equipment
  + Add: Admin can add equipment to his\her co-working space through its attributes name, description like Printer, kitchen etc.
  + Retrieve, Update, delete equipment
* Manage Facilities
  + Add: Admin can add a facility to his\her co-working space through its attributes name, description like skype room, relax zone, parking zone etc.
  + Retrieve, Update, delete facilities
* Manage Offers
  + Add: Admin can add new offers or promotions on his\here place resources like meeting rooms, private offices etc. through its attributes date, start time, end time, discount rate, description.
  + Retrieve, Update, delete offers including expired and coming.
  + Close current available offer.
* Manage Posts
  + Add: Admin can post to public which will be seen by users to facilitate social community.
  + Retrieve, Update, delete Posts filtered by latest posts.
* Manage Reviews
  + View all reviews about the place to listen to the end users issues.
  + Delete reviews
  + Reply to reviewers by sending them an email.
* Manage Opening Times
  + Add: Admin can add new opening time for the place by entering day, from, to data.
  + Retrieve, Update, delete opening times of the place
* Manage Support Issues
  + Reply: Admin can reply to end user complaining about some issues through sending an email to them with solutions.
  + Retrieve, delete support emails history.
* Manage Blocked Users
  + Block User: Admin can block user who book several times and didn’t come from reservation again to his\her co-working space.
  + Unblock User: unblock user after certain amount of time.
  + Retrieve all blocked users.
* Update Profile Settings.
* Disable Co-working Profile.

### System Functions

* Classify booking request
  + Automatically System will classify upcoming booking requests to if really the booking’s user will come or not and showing its probability.
* Show Statistic Charts
  + System gives to the co-working’s owner insights about his place behavior like chart about each month and the corresponding number of bookings in that month.
  + A chart explaining booking requests current year divided monthly.
  + A chart explaining the most booked assets.
  + A chart explaining the most frequent areas people book from.
  + A chart explaining profit of current year divided monthly.
  + A chart comparing profit for current and last year divided monthly.
  + A chart explaining the top 10 booked users.
  + A chart explaining the most frequent booked user’s ages.
* Show nearest co-working Spaces
  + System will show to the user the nearest co-working space that may best fit for him/her.
* Show newest Added Spaces
  + System will show the newest added co-working spaces to users to explore them.
* Calculate Booking Cost

### General Functions

* Sign Up (End User)
  + User can sign up to the system by entering his\her data like name, email, password, address, age, gender.
* Login (End User)
  + User can login to the system by entering his\her email and password.
* Login (Admin)
  + Admin can login to the system by entering co-working’s email and password.
* Logout (End user and Admin)
  + End user and Admin both can logout to the system.

## Non-Functional Requirements

**Usability**

* Our system will be easy to use by choosing simple mock-up, and we will enhance interface by taking feedback from Stakeholders.
* Reach any function in minimum number of clicks (3 for example).
* System will delay for any reason, we will produce message to user telling him to wait.
* Button in interface will have Symbols that show the function of that button.

**Reliability**

* System will handle wrong inputs from stakeholders by 100% every input from user will be checked.
* System should identify the co-working space’s location on map correctly.
* Percentage of failure will be less than 20%.

**Performance**

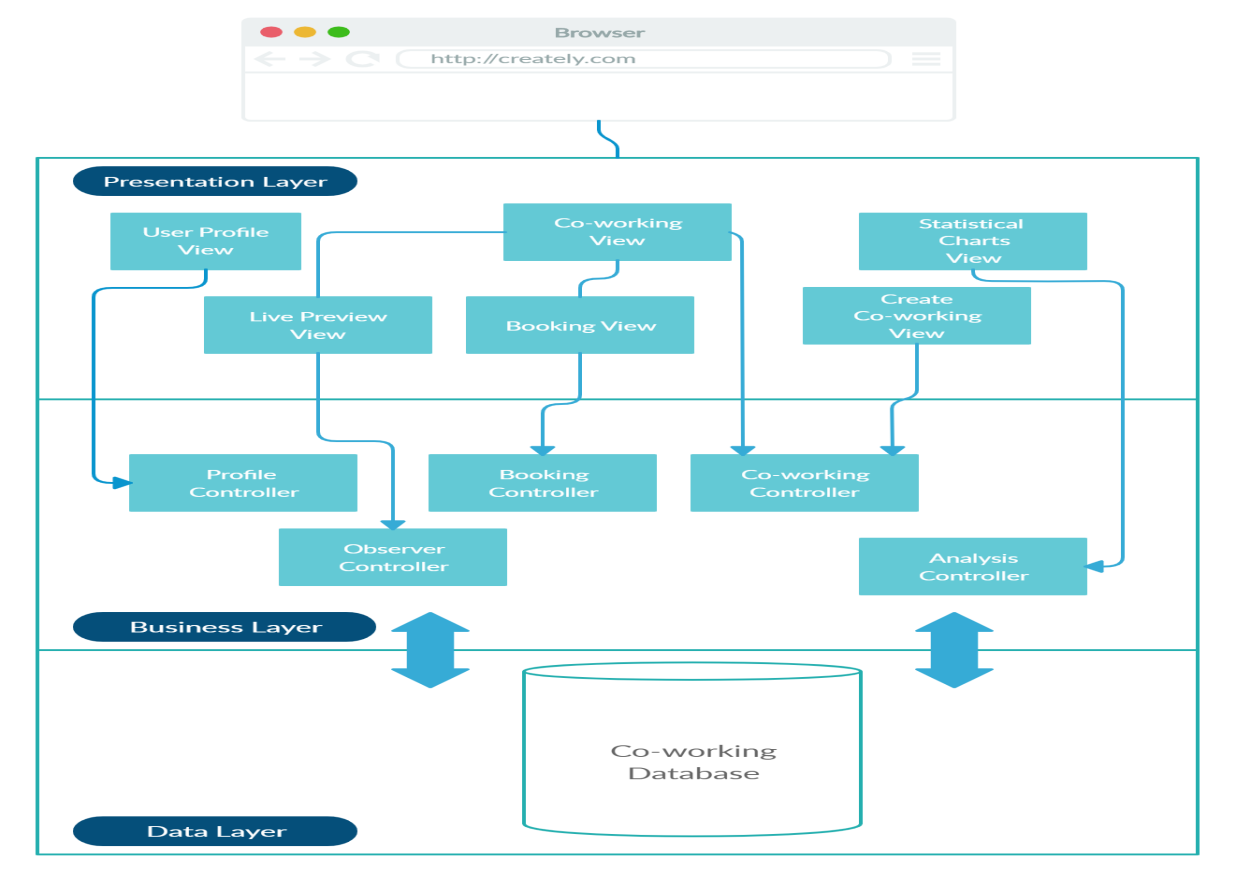
* Response time: maximum operation time will be 1: 30 seconds.
* Scalability: The system will be accept new featured that we would add in future easily and we will cover large scale of places.
* The System uses queues to handle the increasing workloads.
* Availability: The system will be available 24 hours a day.

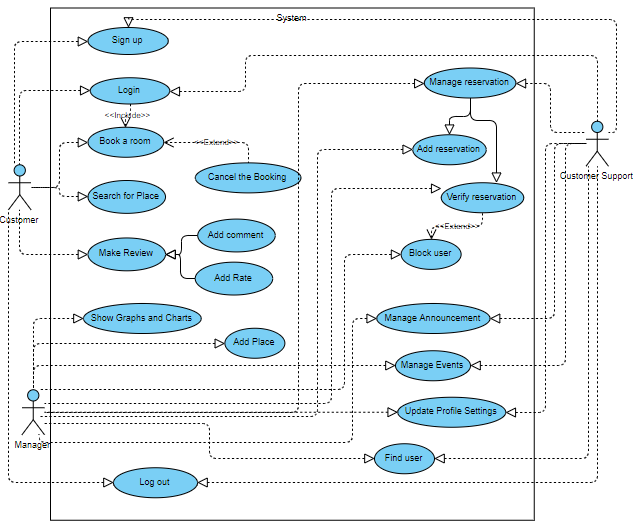
**Supportability**

* System can be adapted to different environments and configurations.
* Programmers can add new features and test it no more than one week.
* System will be down for maintenance less than 24 hour.

# Chapter 3: Requirements Modeling

## System Architecture Diagrams





## Use cases Diagrams

## Use case Tables

### Admin

Login

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Log in. | |
| **Brief Description** | Admin logs in to the System. | |
| **Actor(s)** | Administrator of the place. | |
| **Pre-Conditions** | Already a logged-in admin and have access to this place. | |
| **Post-Conditions** | Logged in to the system successfully. | |
| **Flow Of Events** | **Admin Action** | **System Action** |
|  | 1-Admin enters login information | 2-The system accepts the login information  3-Checks login information in the database. |
| **Exceptions** | **Admin Action** | **System Action** |
|  | 1-Admin enters invalid information | 2- System refuses login process, and requests user to try again. |

Log out

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Log Out. | |
| **Brief Description** | Admin logs out from the System. | |
| **Actor(s)** | Logged in Admin. | |
| **Pre-Conditions** | Already a logged-in admin. | |
| **Post-Conditions** | Logged out from the system successfully. | |
| **Flow Of Events** | **Admin Action** | **System Action** |
|  | 1-Admin clicks on “Log Out”.  3-Admin confirms he/she wants to log out. | 2-System displays a prompt asking if the user indeed wishes to logout.    4-System returns the user to log in screen and the session is deleted. |
| **Exceptions** | **Admin Action** | **System Action** |
|  |  |  |

Manage Reservation ( Add Reservation )

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Add Reservation. | |
| **Brief Description** | The admin adds a reservation. | |
| **Actor(s)** | Administrator | |
| **Pre-Conditions** | Logged in account -> Reservations -> Add. | |
| **Post-Conditions** | A reservation is added successfully. | |
| **Flow Of Events** | **Admin Action** | **System Action** |
|  | 1-Admin clicks on “Reservations ” , then “Add” button.  3-Admin fills in the reservation form based on user input.  6-Clicks on “Done”. | 2- The system shows a reservation form to be filled by admin.  4- System adds this reservation to database.  5- “Done” prompt. |
| **Exceptions** | **Admin Action** | **System Action** |
|  | 1-Admin enters invalid information | 2- System refuses admin’s addition process, and requests admin to try again. |

Manage Reservation ( Add User Extended from Add Reservation)

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Add User. | |
| **Brief Description** | The admin adds user to the system. | |
| **Actor(s)** | Administrator | |
| **Pre-Conditions** | Logged in account -> Users-> Add. | |
| **Post-Conditions** | A user is added successfully. | |
| **Flow Of Events** | **Admin Action** | **System Action** |
|  | 1-Admin clicks on “Users” , then “Add” button.  3-Admin fills in the addition form based on user input.  6-Clicks on “Done”. | 2- The system shows sign up form for user.  4- System adds this user to database.  5- “Done” prompt. |
| **Exceptions** | **Admin Action** | **System Action** |
|  |  |  |

Verify Reservation

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Verify Reservation. | |
| **Brief Description** | The admin verifies a reservation request. | |
| **Actor(s)** | Administrator | |
| **Pre-Conditions** | Logged in account -> Reservations. | |
| **Post-Conditions** | Verify reservation request successfully. | |
| **Flow Of Events** | **Admin Action** | **System Action** |
|  | 1-Admin clicks on “Reservations ” button.  3-In case of true reservation (The users truly came), The admin clicks on “Confirmed” | 2- The system shows a table of reservations (if found).    4- System confirms this reservation. |
| **Exceptions** | **Admin Action** | **System Action** |
|  | 1-In case of False Reservation | 2- Use case # Block User (extended from Verify Reservation) |

Block User (extended from Verify Reservation)

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Block User. | |
| **Brief Description** | The admin block a user with false reservation. | |
| **Actor(s)** | Administrator | |
| **Pre-Conditions** | Logged in account -> Reservations. | |
| **Post-Conditions** | Verify reservation request successfully. | |
| **Flow Of Events** | **Admin Action** | **System Action** |
|  | 1-Admin clicks on “Reservations ” button.  3-In case of false reservation (The users didn’t show up until some time), The admin clicks on “Destroy Reservation” | 2- The system shows a table of expired reservations (if found).    4-System destroys this reservation.  5-System makes this reservation time/room available again. |
| **Exceptions** | **Admin Action** | **System Action** |
|  |  |  |

Manage Facilities(CRUD)

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Manage Facilities. | |
| **Brief Description** | The admin manages Facilities of the place. | |
| **Actor(s)** | Administrator | |
| **Pre-Conditions** | Logged in account -> facilities. | |
| **Post-Conditions** | Facilities is read/added/updated/deleted successfully. | |
| **Flow Of Events** | **Admin Action** | **System Action** |
|  | 1-Admin clicks on “Facilities” tab.  3-If Admin clicked on:  1) Add  2) Update    3) Delete | 2- The system lists all Facilities, with 3 options (add , update , delete).  4-Facilities is added to database and to the list.  5-Facilities is updated in database and to the list.  6-Facilities is deleted from database and from the list.   \*Are you sure? Prompt is shown after each process. |
| **Exceptions** | **Admin Action** | **System Action** |
|  |  |  |

Manage Announcements (CRUD)

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Manage Announcements . | |
| **Brief Description** | The admin manages Announcements of the place . | |
| **Actor(s)** | Administrator | |
| **Pre-Conditions** | Logged in account -> Announcements . | |
| **Post-Conditions** | Announcements is read/added/updated/deleted successfully. | |
| **Flow Of Events** | **Admin Action** | **System Action** |
|  | 1-Admin clicks on “Announcements ” tab.  3-If Admin clicked on:  1) Add  2) Update    3) Delete | 2- The system shows Announcements list, with 3 options (add , update , delete).  4-Announcements is added to database and to the list.  5-Announcements is updated in database and to the list.  6-Announcements is deleted from database and from the list.   \*Are you sure? Prompt is shown after each process. |
| **Exceptions** | **Admin Action** | **System Action** |
|  |  |  |

Manage Events (CRUD)

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Manage Events. | |
| **Brief Description** | The admin manages Events of the place. | |
| **Actor(s)** | Administrator | |
| **Pre-Conditions** | Logged in account -> Events. | |
| **Post-Conditions** | Events is read/added/updated/deleted successfully. | |
| **Flow Of Events** | **Admin Action** | **System Action** |
|  | 1-Admin clicks on “Events ” tab.  3-If Admin clicked on:  1) Add  2) Update  3) Delete | 2- The system shows Events list, with 3 options (add , update , delete).  4-Events is added to database and to the list.  5-Events is updated in database and to the list.  6-Events is deleted from database and from the list.   \*Are you sure? Prompt is shown after each process. |
| **Exceptions** | **Admin Action** | **System Action** |
|  |  |  |

Update Profile Settings

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Update Profile Settings. | |
| **Brief Description** | The admin updates profile settings. | |
| **Actor(s)** | Administrator | |
| **Pre-Conditions** | Logged in account -> Profile Settings. | |
| **Post-Conditions** | Profile settings is updated successfully. | |
| **Flow Of Events** | **Admin Action** | **System Action** |
|  | 1-Admin clicks on “Profile Settings” button.  3- Admin clicks on “Update” button.  5-Admin fills in updates. | 2- System shows profile of place to user with “Update button”.    4-Input field is shown to user.  6-System checks its validity.   7-Adds updates to database. |
| **Exceptions** | **Admin Action** | **System Action** |
|  |  |  |

Find(Search) User

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Find User. | |
| **Brief Description** | The admin finds user. | |
| **Actor(s)** | Administrator | |
| **Pre-Conditions** | Logged in account -> Find User box. | |
| **Post-Conditions** | Retrieve searched user profile. | |
| **Flow Of Events** | **Admin Action** | **System Action** |
|  | 1-Admin enters User ID/Name into the Find User Box. | 2- The system displays results |
| **Exceptions** | **Admin Action** | **System Action** |
|  |  |  |

### Manager

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Add new Co-working space. | |
| **Brief Description** | An administrator of the System adds a new Co-working space to the database. | |
| **Actor(s)** | Administrator / Normal User. | |
| **Pre-Conditions** | Logged in user -> Add new space. | |
| **Post-Conditions** | A new co-working space is added successfully. | |
| **Flow Of Events** | **User Action** | **System Action** |
|  | 1-Login  2-Add new space  4-User adds space’s information | 3-System confirm new space    5- System approves a new space and adds it to the database after checking it's data |
| **Exceptions** | **User Action** | **System Action** |
|  | 1. A user enters invalid information | 2- The system refuses to add process, and requests user to try again. |

### User

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Sign Up. | |
| **Brief Description** | A user of the system creates an account. | |
| **Actor(s)** | Guest. | |
| **Pre-Conditions** | Internet-connected and Opening website/app. | |
| **Post-Conditions** | A new account is created successfully. | |
| **Flow Of Events** | **User Action** | **System Action** |
|  | 1-User clicks on Create new account  3-User enters registration information  The 5-Home page is shown to the user. | 2-System confirms a new user  4- System approves a new user and adds it to the database after checking it's info. |
| **Exceptions** | **User Action** | **System Action** |
|  | 1-User enters invalid information | 2- The system refuses to create a new account process, and requests user to try again. |

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Log in. | |
| **Brief Description** | A user of the System logs in to the System. | |
| **Actor(s)** | Logged in user. | |
| **Pre-Conditions** | Already a logged-in user. | |
| **Post-Conditions** | Logged in to the system successfully. | |
| **Flow Of Events** | **User Action** | **System Action** |
|  | 1-User enters to log in info.  The 4-Home page is shown to the user. | 1. The system accepts the login info. 2. Checks log in info in the database. |
| **Exceptions** | **User Action** | **System Action** |
|  | 1-User enters invalid information | 2- System refuses to log in process, and requests user to try again. |

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Log Out. | |
| **Brief Description** | A user of the System logs out from the System. | |
| **Actor(s)** | Logged in user. | |
| **Pre-Conditions** | Already a logged-in user. | |
| **Post-Conditions** | Logged out from the system successfully. | |
| **Flow Of Events** | **User Action** | **System Action** |
|  | 1-User clicks on Log Out.  3-User confirms he/she wants to log out. | 2-System displays a prompt asking if the user indeed wishes to logout.    4-System returns the user to log in screen and the session is deleted. |
| **Exceptions** | **User Action** | **System Action** |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Search. | |
| **Brief Description** | A user searches about a place. | |
| **Actor(s)** | Guest/ Logged in user. | |
| **Pre-Conditions** |  | |
| **Post-Conditions** | Showing results matching his/her preferences/filters. | |
| **Flow Of Events** | **User Action** | **System Action** |
|  | 2-User enters search criteria and submits. | 1-The system displays the search submission box  3-The system displays results |
| **Exceptions** | **User Action** | **System Action** |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Book. | |
| **Brief Description** | A user wants to book a chair/room/area or whatever. | |
| **Actor(s)** | Logged in user. | |
| **Pre-Conditions** | Logged in user. | |
| **Post-Conditions** | The booking process is done successfully. | |
| **Flow Of Events** | **User Action** | **System Action** |
|  | 1-User enters some place’s profile on the system.  3-User clicks on the "Book" button.    5-User enter booking details.  8-User confirms the booking. | 2-System shows the place's profile to the user.  4- System displays a prompt asking users to enter more booking details.  6-System checks entered booking info.  7-System asks the user to confirm the booking.  9-System adds booking details in the database and assign it to that user. |
| **Exceptions** | **User Action** | **System Action** |
|  | 1-User is not logged in to the system.      3- If the user is already logged in, enter login info.  4- If not, he/she is redirected to Sign up page. | 2-System displays a login prompt and "don't have an account yet"  5-System redirects the user to Sign up page. |

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Add comment | |
| **Brief Description** | A user of the System adds a new comment on a specific co-working space | |
| **Actor(s)** | User | |
| **Pre-Conditions** | Logged in user -> search about place -> open profile -> add comment | |
| **Post-Conditions** | A new comment is added successfully | |
| **Flow Of Events** | **User Action** | **System Action** |
|  | 1-Login  2-Search about place  4-User opens the place profile  6-User click to add a comment and enter his words | 3-System shows a list of results  5-The system shows the place details    7-System add his comment successfully |
| **Exceptions** | **User Action** | **System Action** |
|  | 1. User doesn’t enter words and need to add his comment | 2- System make the add button disabled until the user enters words. |

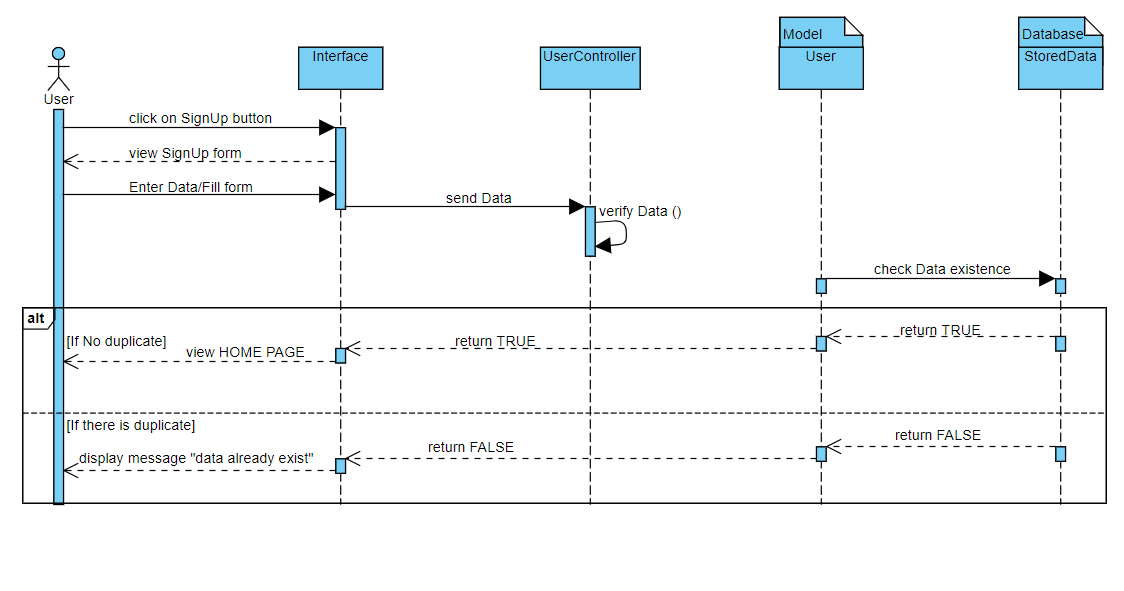
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Add review | |
| **Brief Description** | A user of the System adds a new comment on a specific co-working space | |
| **Actor(s)** | User | |
| **Pre-Conditions** | Logged in user -> search about place -> open profile -> add review | |
| **Post-Conditions** | A new review is added successfully | |
| **Flow Of Events** | **User Action** | **System Action** |
|  | 1-Login  2-Search about place  4-User opens the place profile  6-User click to add review | 3-System shows a list of results  5-System shows the place details    7-System adds his review successfully |
| **Exceptions** | **User Action** | **System Action** |
|  | 1-User doesn’t enter words and need to add his comment | 2- System make the add button disabled until the user enters words. |

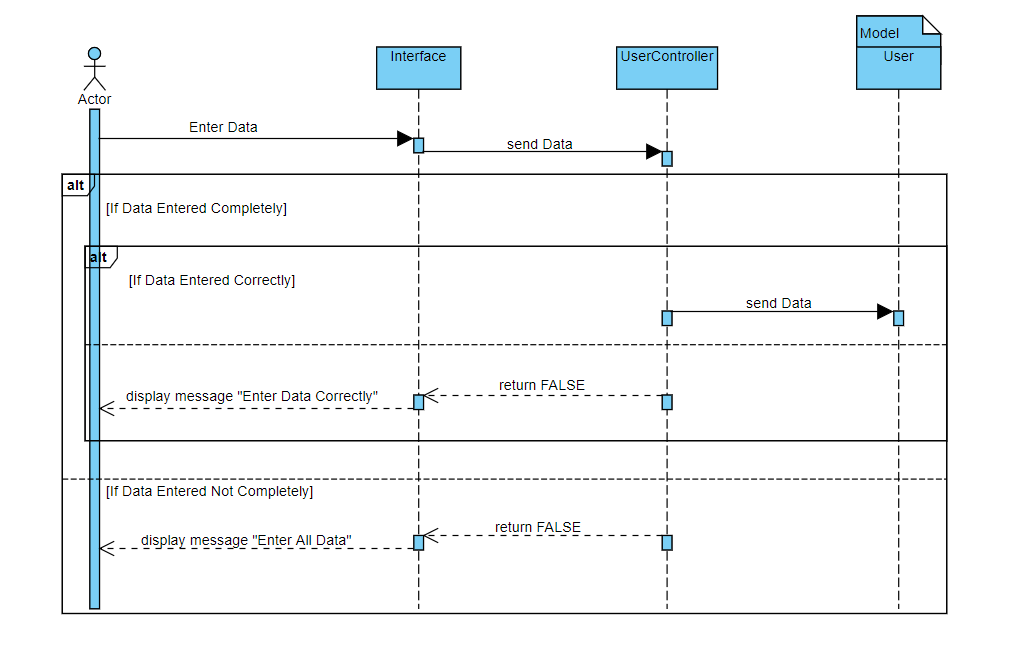
|  |  |  |
| --- | --- | --- |
| **Use Case ID** | # | |
| **Use Case Name** | Cancel booking | |
| **Brief Description** | A user of the System adds a new comment on a specific co-working space | |
| **Actor(s)** | User | |
| **Pre-Conditions** | Logged in user -> search about place -> open profile -> add review | |
| **Post-Conditions** | A new review is added successfully | |
| **Flow Of Events** | **User Action** | **System Action** |
|  | 1-Login  2-Enter his profile and choose bookings from the menu list  4-User click on the booking that he wants to cancel and click cancel | 3-System shows the list of bookings    5-System remove that booking |
| **Exceptions** | **User Action** | **System Action** |
|  | 1-User need to cancel a booking after the valid period for making this action had been passed | 2- System displays a message  According to reservation policy |

## 86266504_500096910697309_90373864323809280_n System Context Diagram

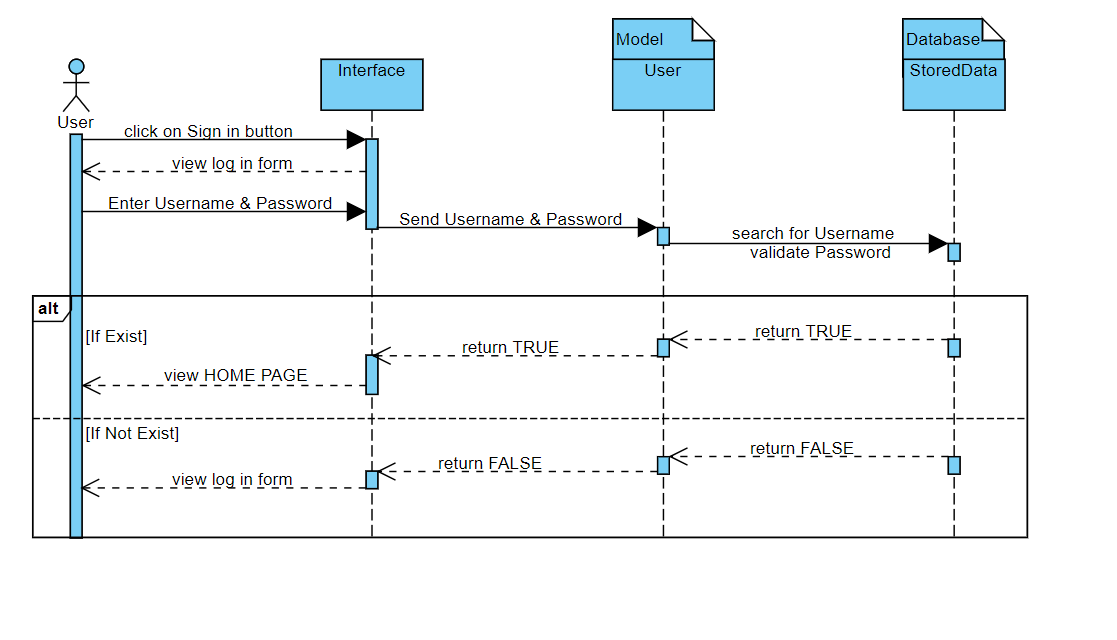
## Sequence Diagram

### Sign up

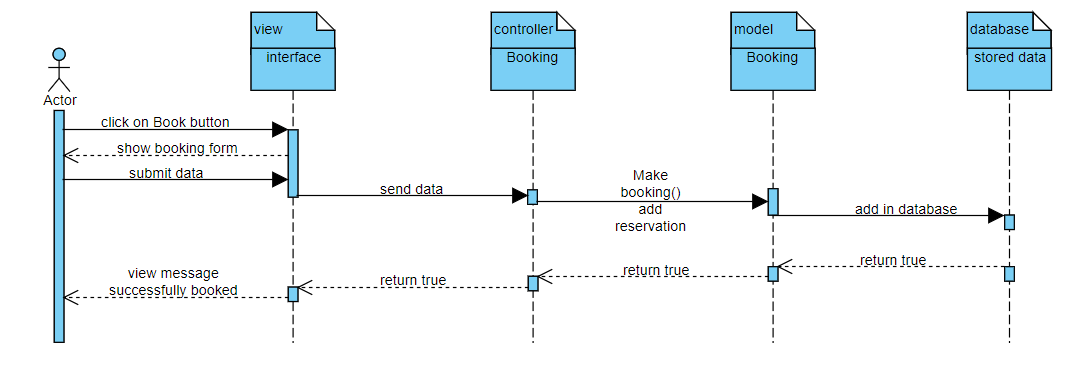




### Login



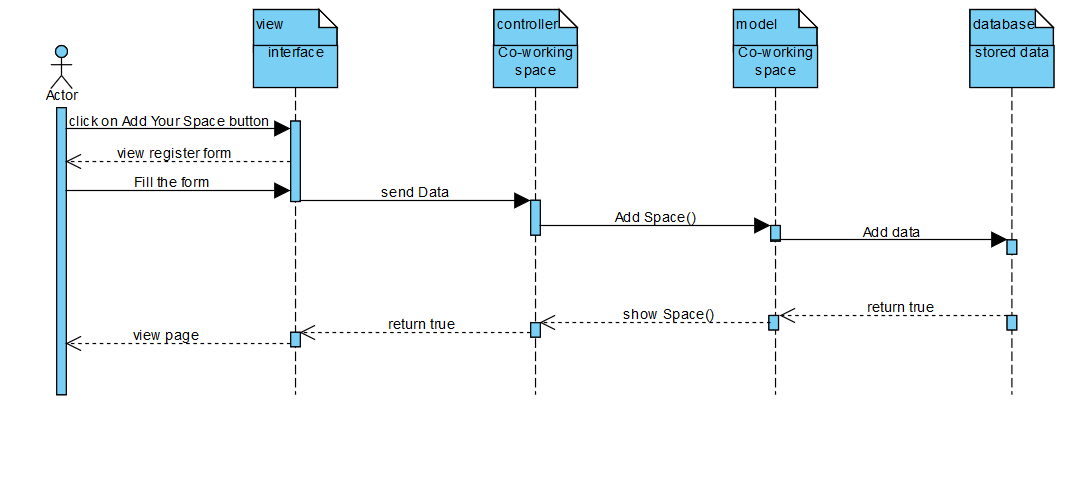
### Booking



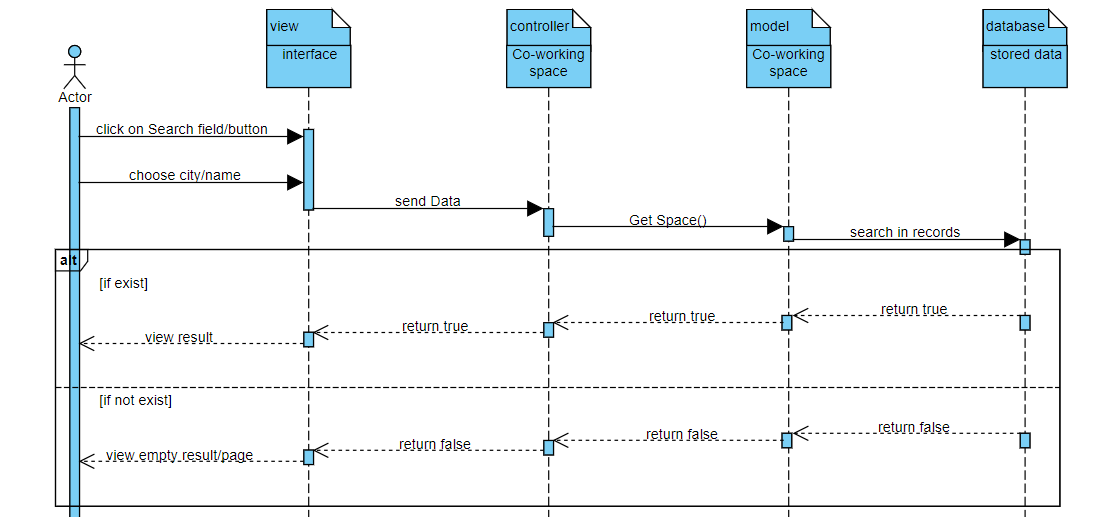
### Booking cancellation



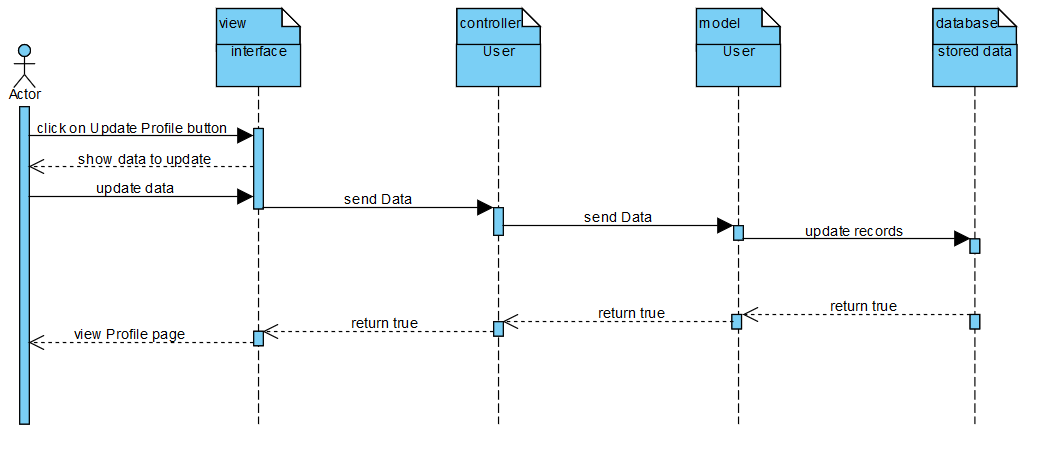
### Add co-working space



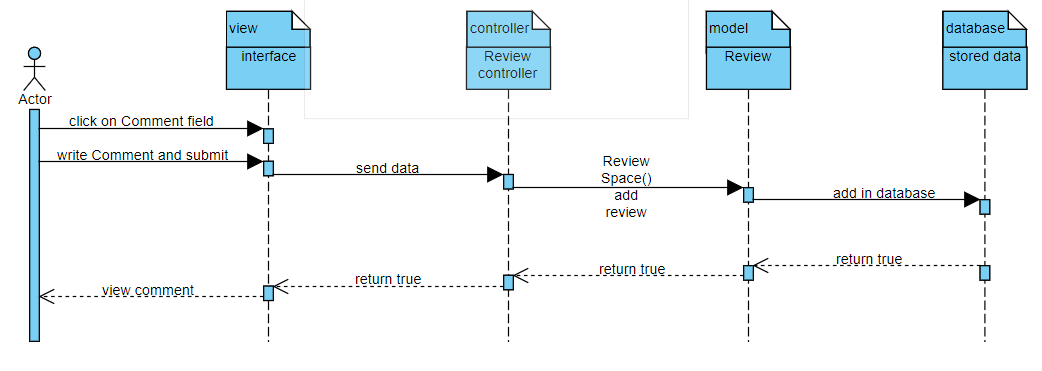
### Search for place



### Update profile



### Add review /comment



## class diagramClass Diagram

## 

## ERDEntity relationship Diagram

# Chapter 4: Design and Issues:

## Covering tools / platforms used:

* Mobile Development:
  + Android Studio, Visual Studio,
* Web Development:
  + Brackets, PHP Storm, Sublime, Visual Studio
* Other:
  + GitHub
  + Trello
  + Zoom

## Issues (Challenges)

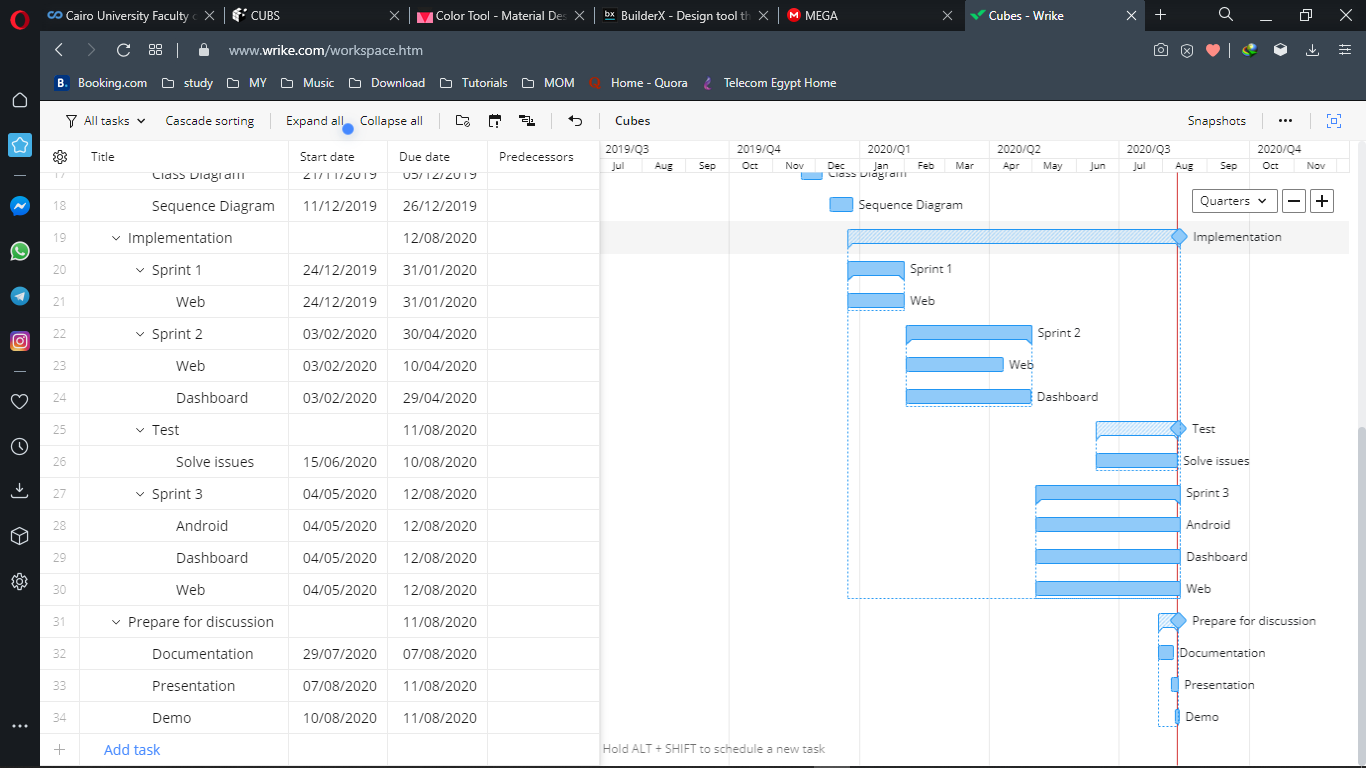
* Lack of sufficient Data – needed for Data Mining.
* Lack of resources/tutorials of some frameworks which lead us to abandon it and find an alternative ex: Flutter, …
* Lack of free online servers to upload our work on it.
* Because of lockdown, we couldn’t compile work at one place.
* Choosing right and suitable framework.
* Connecting backend API with Mobile.
* Because of Corona and lockdown, we couldn’t perform some types of tests as:
  + acceptance test

# Chapter 5: System Testing and Evaluation:

## **Testing:**

Originally test plan was put in the very beginning of the project, in fact it is divided into parts, first one is local testing; which is done among us as: unit testing, then integration testing when we compile work, System testing when all the parts are gathered together, also let’s not forget the testing done with our Dr and TA , who were considered as our backend users and consultants over the year**.** Second one is global testing I.e. test on real world (real users) as acceptance testing, but unfortunately, we couldn't do that as we’ve mentioned above, but since we communicated with real users from real world as we’ve talked in Chapter 2: Problem Analysis that made us nearly close to end users and understand their needs not ours.

## **Planning**



## **Lesson Learnt**

### Technical Skills:

* PHP
* Laravel
* Mysql
* Javascript
* Restful API
* HTML5, CSS3, SCSS
* Xampp server
* React Native

### Non-Technical Skills:

* Communication skills
* Documenting skills
* Work remotely from each other
* Adapt to circumstances
* Get familiar with how work is done in market
* Work in Mini company
* Requirements Analysis
* Data gathering techniques
* Get familiar with Agile methodology and following it
* Get familiar with sprints concepts
* Team working
* Tasks distribution
* Time planning
* Planning
* Testing techniques
* Problem solving
* Decision making

## **Future work**

* Online Payment:
  + Pay online with different methods
  + Paypal
  + Credit card/Visa
  + Fawry
* Recommendation Module:

Willing to build and develop efficient and valuable recommendation module that adds a value to the existing solution and provides satisfied customer experience for system users throw their behavior and catching every action from them as a result recommending the best fit co-working spaces for a user.

* Deploying Solution:

Willing to go further with the last step by deploying the developed solution to production on a live server to be accessed by public.

* User location detection:

Based on GPS technology, System will detect the nearest co-working space to the current user location and notify user regarding available offers at that co-working space.