# Software Requirement Specification Document for Jardin

Mahy Ayman, Mariam Othman, Nour Mahmoud, Zeina Tamer

May 7, 2021

Table 1: Document version history

| **Version** | **Date** | **Reason for Change** |
|---|---|---|
| 1.0 | 28-April-2021 | SRS First version's specifications are defined. |

**GitHub:** https://github.com/nourelgarhyy/Jardin/projects/1

# Contents

**Abstract**

Coworking spaces are becoming increasingly common among freelancers, information workers, and start-up communities in modern cities. Coworking spaces can become a burden sometimes, one can never know when it is the right time to go there since individuals can never know if it'll be too crowded or if there will be any room left for one to complete their work in. Together with our web application, the coworking space could provide a more controlled environment than that one might find in a cafe. The role of our web application in solving this problem is to ensure a safe way of communication between freelancers/students/people and the coworking space. The development process is mainly built on making communication much easier, and safer to establish. The proposed solution is a web application that will facilitate people's communication with our clients' co-working space. Our web application aims to ensure a seamless experience from landing on the web application until physically landing on the place itself. It will be easier to reserve or contact the coworking space without going through phone calls or other procedures that may cause loss of time, and for some people anxiety.

# 1 Introduction

## 1.1 Document Purpose

The purpose of the document is to conceptualize the project and provide detailed requirements for the development of Jardin web application. It will also provide a a brief overview of the various functions of the web application. The document will define the functional and non-functional requirements. The document will also be beneficial to anyone who will work on future maintenance of the web application.

## 1.2 Document Scope

The project is developing a web application that would be considered as the one-stop shop for Jardin. It will have all the services provided by Jardin. This web application helps with creating the brand identity. One of the main objectives of the web application is to act as a communication network between managers and their guests. The goal of this project is to provide a systematic and automated reservation system.

## 1.3 Business Context

Jardin is an outdoor coworking space that has come to life as a result of the changes that has taken place in the world over the last year due to COVID-19. It offers rooms and spaces that guests could reserve for a few hours or a full day to work or host meetings as well as organized events for its guests and community members for entertainment.

## 2 Similar Systems

### 2.1 Academic

Students in computer science department in Nigeria  [1]. , developed an online booking system for cinema house. The work analyzed the introduction of e-services which shows that e-commerce website can promote a trendy way for people to perform booking/reservation and also suggested online booking systems can be developed for bus stations, airports, hotels, cinemas, coworking spaces and other centers that engage in reservation. The researchers used Hypertext MarkUp Language, Cascading Style Sheet and JavaScript for the front end and MySQL database as the back end; and PHP as the scripting language.  The paper however failed to mention any obstacles that the researchers has gone through either in implementation nor in business field.

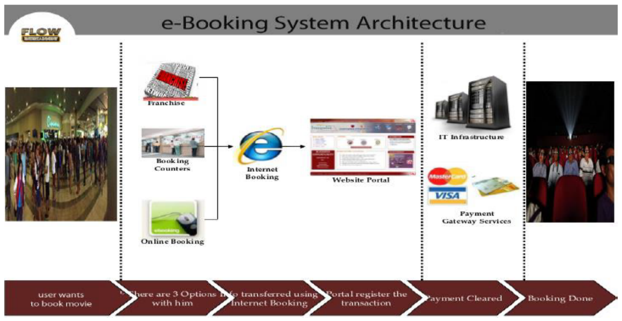
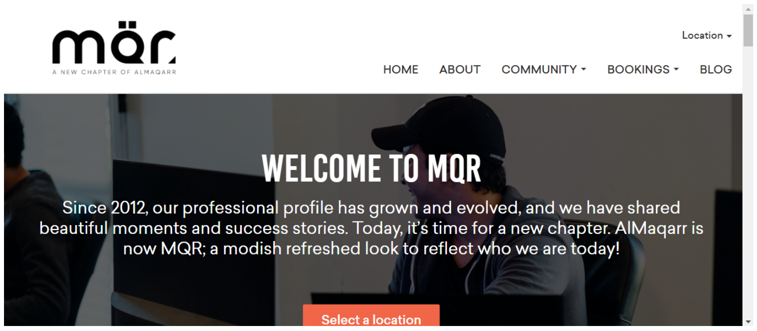


Figure 1: The System architecture of the proposed system

## 2.2 Business Applications

Similar applications revealed that a number of websites search for several co-working places at once and are not dedicated to a single one. This concept is good for searching for any random offices around the client, but it struggles to create a direct contact between the clients and each office. One of the few dedicated web applications we found was that of the 'MQR' coworking space, however the drawback it has is that the user is forced to signup to view content which creates a poor user experience. Another drawback is that the user might get lost eventually in the web application because of the number of clicks one clicks. See Figure 2.



Since 2012, our professional profile has grown and evolved, and we have shared beautiful moments and success stories. Today, it's time for a new chapter. AlMaqarr is now MQR; a modish refreshed look to reflect who we are today!

Figure 2: MQR is a coworking space

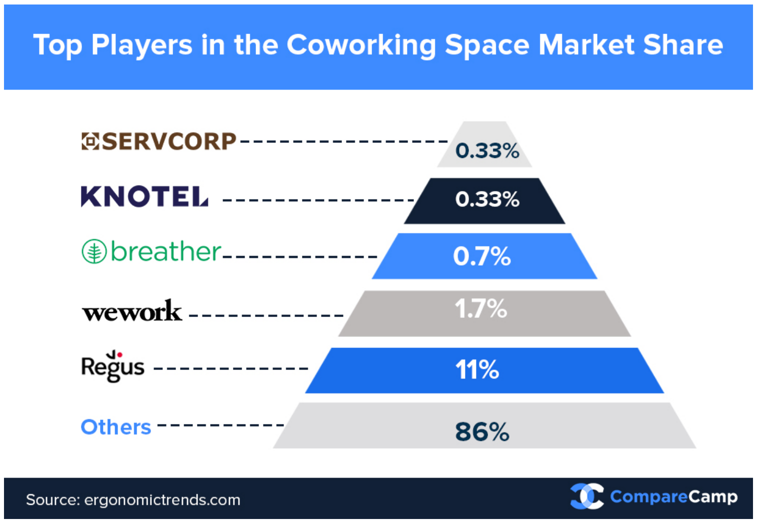


Figure 3: existing applications

# 3 System Description

## 3.1 Problem Statement

People nowadays tend to feel safer in their homes more than any other place due to the situation that has been going on for over a year now with COVID-19. Most people now work, study, and take classes from home, but that could sometimes be boring and mentally exhausting for people who miss their old lives. Coworking spaces have been existing for quite some time now; going to a coworking space is a simple form of change if someone wants to work or study in any other environment other than his/her house. Nevertheless, a coworking space is usually a closed area, and that could be problematic these days.

## 3.2 System Overview

By following through with the requirements of our clients, our product should be an easily navigated web application that offers its users a brief look to what they should expect on visiting the co-working space including schedules, calendar of important events and the offered accommodations. Moreover it should also work as an open channel of communication between the hosts and their community of registered visitors and a fully functioning booking system.

## 3.3 System Scope

- The web application is expected to offer its users a glimpse of what to expect at Jardin such as events and to offer details about the spaces available.
- The web application would act as a base for the community members and registered users to communicate with each other and with the managers and offer online reservations.
- Provide regular reports and statistics of users and their ratings, services.

## 3.4 System Context

The software interacts with employees, managers and of course the customers. In addition, during the development process a number of laws, norms, guidelines and style guides will have an impact on the final design of the system.

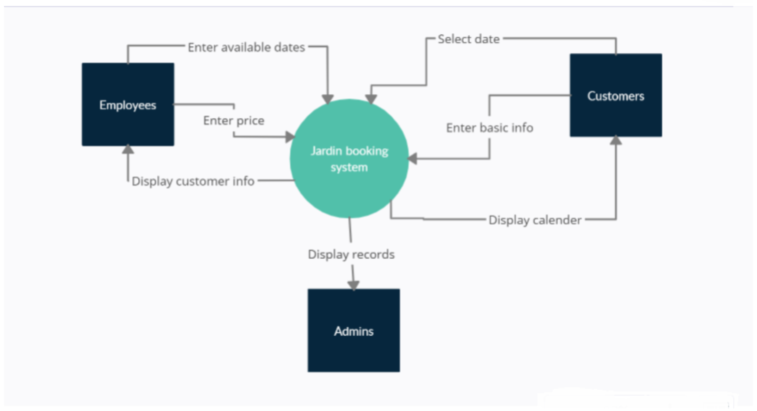


Figure 4: Context Diagram

## 3.5 Objectives

Creating a medium between the owners and clients for taking requests with loading at most 3 pages.
• Users should find what they need in least amount of clicks by the end of the design period.
• Users should view the basic services without having to sign-up, it's only when they're reaching the last step which is the reservation they'll have to sign-up.
• Users should get the feeling of wanting to signup after using the basic features of the web application rather than having to signup.

## 3.6 User Characteristics

Our users are mostly everyday people with the average expertise with software systems. We are targeting ages starting from early 20s to 40s, with various job responsibilities but mainly ones that could be carried out out of office or educational jobs that require open spaces to carry out activities. The company internal users would be the owners of Jardin who also possess average computer knowledge.

## 3.7 User Stories

As a user, I would like to be able to browse all the pages of the web application without having to register/log in first and to only have to do that once it's necessary (such as booking a space/sending a message to admins). I would like to be able to access most pages from the homepage. I want to receive an email reminder of my reservations and alerts of upcoming events/ activities. As a manager, I want to be able to view user complaints/reviews/statistics of surveys. I would like to receive a monthly average of the ratings left after each visit to assess quality offered.

# 4 Functional Requirements

## 4.1 System Functions

1-The user should be able to start a conversation via chat with admins or other community members.

2-The user should be able to make a reservation for a room/space.

3-The user should be able to view calendar with pre-scheduled events.

4-The admin should be able to add/remove events.

5-The user should be able to register/join the online community.

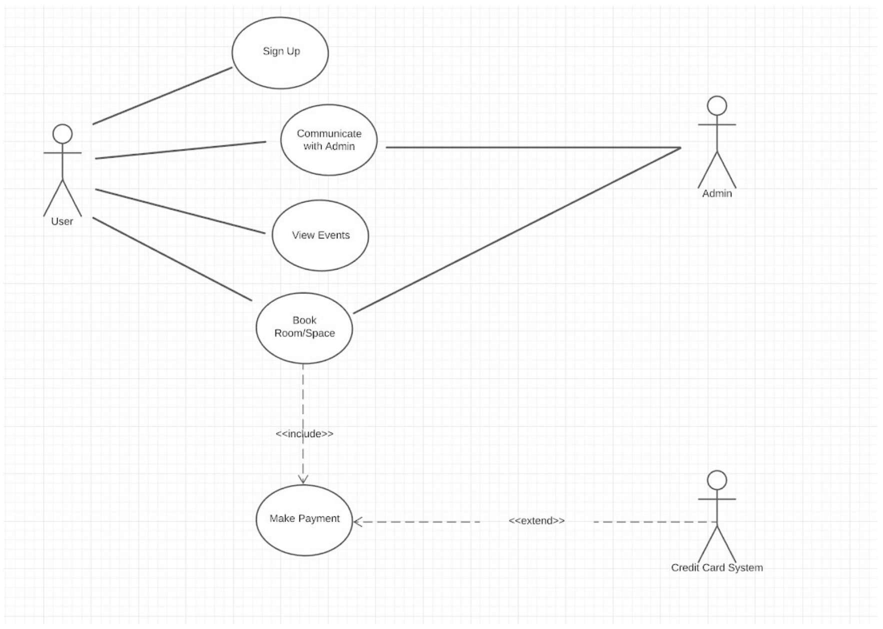


Figure 5: Case Diagram

## 4.2 Detailed Functional Specification

Table 2: Add Event Function Description

| | |
|---|---|
| **Name** | addEvent |
| **Code** | AE01 |
| **Priority** | High |
| **Critical** | 8/10. |
| **Description** | Events are added by admin and saved in database. |
| **Input** time, description. | Event name, date |
| **Output** | Void |
| **Pre-condition** | Only admin accounts have access to use this function. |
| **Post-condition** | Update the calendar with new Event. |
| **Dependency** | SI01 |
| **Risk** | No risk. |

Table 3: Remove Event Function Description

| | |
|---|---|
| **Name** | removeEvent |
| **Code** | RE02 |
| **Priority** | High |
| **Critical** | 8/10 |
| **Description** | Events are removed by admin and deleted from database. |
| **Input** | Event name and ID. |
| **Output** | Void |
| **Pre-condition** | Only admin accounts have access to use this function. |
| **Post-condition** | Update the calendar without Event. |
| **Dependency** | SI01 |
| **Risk** | No risk. |

Table 4: Registration Function Description

| | |
|---|---|
| **Name** | register |
| **Code** | UR01 |
| **Priority** | Extreme |
| **Critical** | 8/10 |
| **Description** | Takes data from user and saves it in database. |
| **Input** | username, email, password. |
| **Output** | Void |
| **Pre-condition** | no pre-condition. |
| **Post-condition** | Redirect to view homepage. |
| **Dependency** | Doesn't depend on other requirements. |
| **Risk** | If the same username is already taken, there might occur a problem with the signing up process. We have to add a validation method on registration. |

Table 5: Chatting System Function Description

| | |
|---|---|
| **Name** | chat |
| **Code** | CH |
| **Priority** | High |
| **Critical** | 7/10 |
| **Description** | messages are sent by one user and viewed by the other. |
| **Input** | Message (string). |
| **Output** | Message (string). |
| **Pre-condition** | User must be logged in. |
| **Post-condition** | Content is displayed in chat for users. |
| **Dependency** | UR01, SI01 |
| **Risk** | Timings could cause error in the sequence of the message. |

Table 6: Reservation Function Description

| | |
|---|---|
| **Name** | book |
| **Code** | BK |
| **Priority** | High |
| **Critical** | 8/10 |
| **Description** | Makes the time slot and space unavailable after being booked by a certain user. |
| **Input** | Preferred time-slot and room/space. |
| **Output** | Confirmation for reservation. |
| **Pre-condition** | User must be logged in and space/time-slot are available. |
| **Post-condition** | Redirect to payment method page. |
| **Dependency** | UR01, SI01 |
| **Risk** | Two people might try to book the same space and time slot at the same time. There should be a delay between booking and confirmation for validation. |

Table 7: Add Event Function Description

| | |
|---|---|
| **Name** | signIn |
| **Code** | SI01 |
| **Priority** | High |
| **Critical** | 8/10 |
| **Description** | User signs in to make a reservation and to communicate with members. |
| **Input** | Email and password. |
| **Output** | Signed in successfully. |
| **Pre-condition** | User has to have registered. |
| **Post-condition** | Redirect to homepage. |
| **Dependency** | UR01 |
| **Risk** | Database malfunction. |

Table 8: Add Event Function Description

| | |
|---|---|
| **Name** | editProfile |
| **Code** | EP01 |
| **Priority** | Low |
| **Critical** | 6/10 |
| **Description** | User gets to update their display name, password and profile picture . |
| **Input** | New display, password, profile picture. |
| **Output** | Profile updated successfully. |
| **Pre-condition** | User has to be signed in. |
| **Post-condition** | Show updated profile. |
| **Dependency** | UR01, SI01 |
| **Risk** | Database malfunction. |

Table 9: Add Event Function Description

| | |
|---|---|
| **Name** | logOut |
| **Code** | LO01 |
| **Priority** | High |
| **Critical** | 8/10 |
| **Description** | User signs out of their account. |
| **Input** | User interaction. |
| **Output** | Signed out successfully. |
| **Pre-condition** | User is logged in. |
| **Post-condition** | Redirect to homepage. |
| **Dependency** | SI01 |
| **Risk** | Database malfunction. |

Table 10: Add Event Function Description

| | |
|---|---|
| **Name** | editEvent |
| **Code** | EE03 |
| **Priority** | High |
| **Critical** | 7/10 |
| **Description** | Events are edited by admin and saved in database. |
| **Input** time, description. | Event new name or date |
| **Output** | Event updated. |
| **Pre-condition** | Only admin accounts have access to use this function. |
| **Post-condition** | Update the calendar with updated Event. |
| **Dependency** | SI01, AE01 |
| **Risk** | Database malfunction. |

Table 11: Add Event Function Description

| | |
|---|---|
| **Name** | viewCalendar |
| **Code** | VC01 |
| **Priority** | High |
| **Critical** | 8/10 |
| **Description** | Events and free time slots are viewed by users. |
| **Input** | User interaction. |
| **Output** | Full calendar. |
| **Pre-condition** | any user could view calendar. |
| **Post-condition** | View events’ calendar. |
| **Dependency** | AE01 |
| **Risk** | Database malfunction/Event mixup. |

Table 12: Add Event Function Description

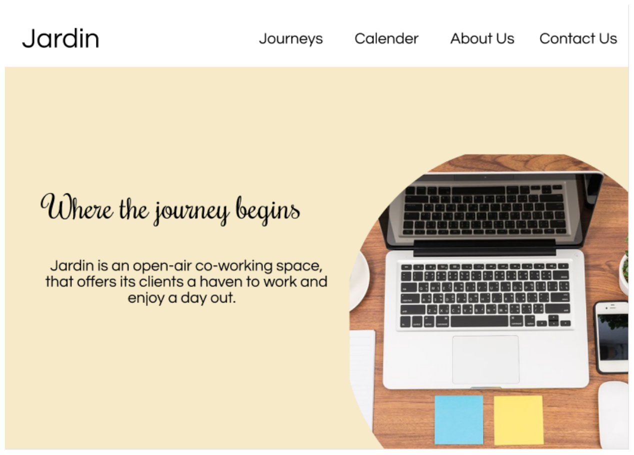
| | |
|---|---|
| **Name** | contactUs |
| **Code** | CU02 |
| **Priority** | High |
| **Critical** | 8/10 |
| **Description** | Non-users could contact admins through this function. |
| **Input** | Email, Message. |
| **Output** | Message sent successfully. |
| **Pre-condition** | Non-registered users could send messages. |
| **Post-condition** | Messages delivered to admins. |
| **Dependency** | No dependency. |
| **Risk** | Database malfunction. |

# 5 Interface Requirements

## 5.1 User Interfaces

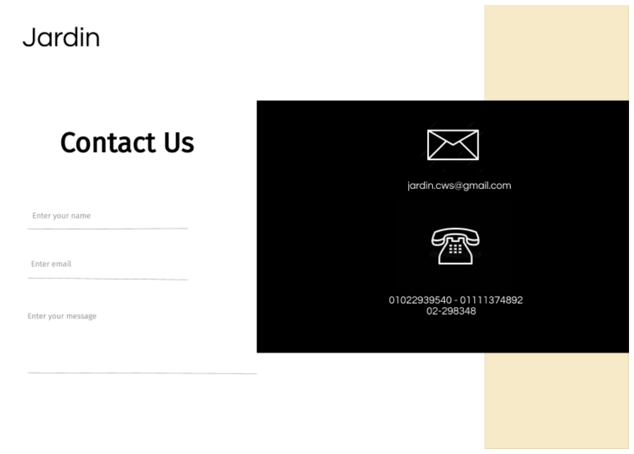
We’re hoping to create a user-friendly web application, in which the user can interact with the interface to reserve, and be aware of the co-working space’s schedule easily and without the hassle of phone calls.

### 5.1.1 GUI



Jardin is an open-air co-working space, that offers its clients a haven to work and enjoy a day out.

Figure 6: Homepage



jardin.cws@gmail.com

01022939540 - 01111374892
02-298348

Figure 7: Contact Us

### 5.2 Communications Interfaces

We're hoping to create a user-friendly web application, in which the user can interact with the interface to reserve, and be aware of the co-working space's schedule easily and without the hassle of phone calls.

## 6 Non-functional Requirements

### 6.1 Security

- The software must ensure the integrity of the customer account information.
- The system shall ensure that the customers data can be viewed only by the authorized users.
- The system shall ensure that the the reservation details are only available to authorized users.

### 6.2 Accessibility

- User will be taken through a smooth experience to know the different journeys the place provide, and that is through pictures and clicks that lead to that.
- Images must have an alt text.
- Forms shall be designed carefully so that each field is clearly labeled and clear validation messages are provided.

### 6.3 Availability

- Users can depend on the system to be up .
- Users can access the system most of the time without failure all days of the week.
- Any scheduled system maintenance should not take more than one day and an alternative page should appear telling the user how long will it take for maintenance to be over. Also, the contact information, like a phone number and email, will be available for users.

### 6.4 Performance

- The home page's load time should not be more than one second for end users.

## 7 Data Design

### 7.1 Data Description

The original format of data should be a simple text file (.txt) or (.docx) for maximum compatibility with future software. Title of data or (DatasetName) should be the title of the data-set and is used to distinguish the data description document from the others in the repository. The data will be captured into the system by importing the (.txt) or (.docx) files. Each data list is expected to have 4 columns (General Overview, Content description, Technical description and Access) and 18 rows. The expected number of customers for the system is about 50-100 each month. [?] The entity key consists of the kind of the entity, which is usually the name of class where the entity exists, and an identifier which is specified by datastore which automatically generate an unused integer numeric ID. The data contains time and date which will be presented in dd-mmm-yyyy (DATEw format). The system will include SQL.

## 7.2 Database design description

The database design distributes data into number of entities based on specific subject. Each entity will become a separate table which models its attributes and its relationship with other tables using one-to-one, one-to-many, and/or many-to-many. Each entity should have its own primary key. Larger tables are divided into smaller tables and are linked together using relationships.

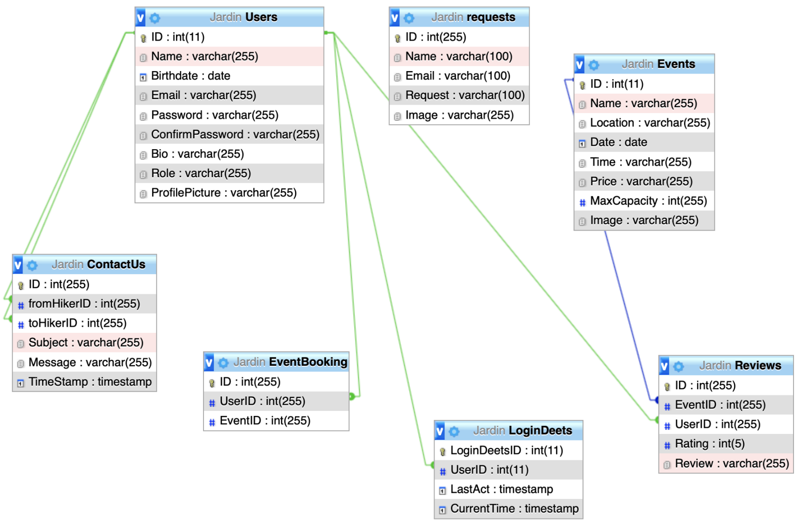


Figure 8: Inheritance Relations

## 8 Preliminary Object-Oriented Domain Analysis

### 8.1 Inheritance Relationships

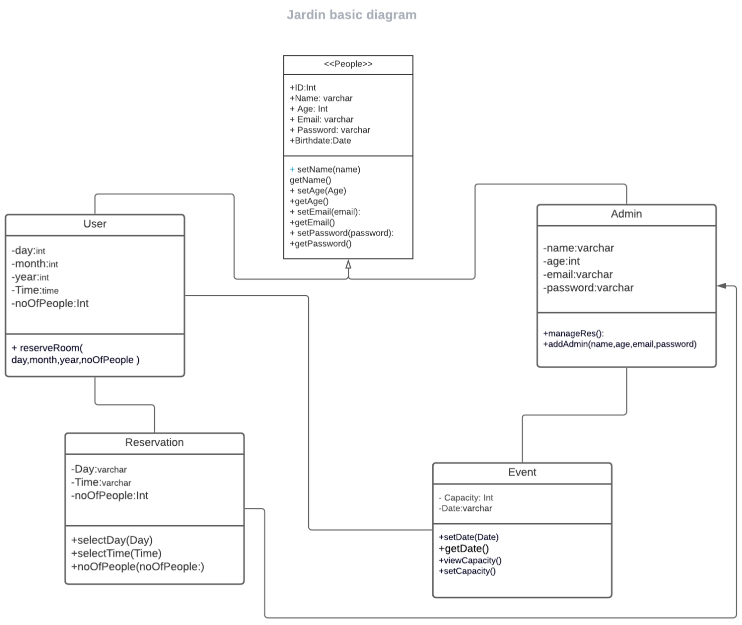


Figure 9: Inheritance Relations

## 8.2 Class descriptions

Table 13: Class Name - People

| | |
|---|---|
| **Abstract or Concrete:** | Abstract Class. |
| **List of Superclasses** | N/A. |
| **List of Subclasses** | User and Admin Classes. |
| **Purpose** | This class has the basic attributes and functions for any person who'll be using the web application. |
| **Collaborations** | Other class mainly inherit from that class, and those classes are the class "User" and the class "Admin". |
| **Attributes** | ID(int), Name(varchar), Age(int), Email(varchar), Password(varchar), Birthdate(date). Example: ID=1, Name=John, Age=28, Email=john123@gmail.com, Password=123456, Birth-date=4/4/1993 |
| **Operations** | setName(name), returns null; getName(), returns name; setAge(age), returns null; getAge(), returns Age; setEmail(email), returns null; getEmail(),returns Email; setPassword(password), returns null; getPassword(), returns Password. |
| **Constraints** | Age and birthate must align together. The email must have a valid email format. Password must not be less than 6 characters. |

Table 14: Class Name - User

| | |
|---|---|
| **Abstract or Concrete:** | Concrete. |
| **List of Superclasses** | People. |
| **List of Subclasses** | N/A. |
| **Purpose** | User class is a subclass from People, and it has some more specific attributes and operations related to the users themselves. |
| **Collaborations** | This class interacts with the "Reservation" class for reservations to happen and this is done using the operations in Reservation class. This class also interacts with the "Events" class to access the event date and details. |
| **Attributes** | day(int), month(int), year(int), Time(time), noOfPeople(int). |
| **Operations** | reserveRoom( day,month,year,noOfPeople), returns null. |
| **Constraints** | If the number of people exceeds the capacity, the reservation will not take place, and the user will be notified. |

Table 15: Class Name - Admin

| | |
|---|---|
| **Abstract or Concrete:** | Concrete. |
| **List of Superclasses** | People. |
| **List of Subclasses** | N/A |
| **Purpose** | Admin class is a subclass from People, and it has some more specific attributes and operations related to the admins themselves and their authority. |
| **Collaborations** | This class interacts with the "Reservation" class. This class also interacts with the "Events" class to set the event date and details. |
| **Attributes** | adminName(varchar), adminAge(int), adminEmail(varchar), adminPassword(varchar), adminBirthdate(date). |
| **Operations** | manageRes(), returns null; addAdmin(adminNAme,adminAge,adminEmail,adminPassword,adminBirthdate), returns null. |
| **Constraints** | Email should be in mail format. Email should bee unique. Password must not be less than 6 characters. |

Table 16: Class Name - Reservation

| | |
|---|---|
| **Abstract or Concrete:** | Concrete. |
| **List of Superclasses** | N/A. |
| **List of Subclasses** | N/A |
| **Purpose** | This class is mainly for managing reservations |
| **Collaborations** | This class is associated with both "User" and "Admin" classes. |
| **Attributes** | day(date), Time(time), noOfPeople(int). |
| **Operations** | selectDay(Day), reutuns null; selectTime(Time), reuturns null; noOfPeople(noOfPeople), reuturns null; |
| **Constraints** | No overlapping reservations should occur, and reservations must not exceed capacity. |

Table 17: Class Name - Event

| | |
|---|---|
| **Abstract or Concrete:** | Concrete. |
| **List of Superclasses** | N/A. |
| **List of Subclasses** | N/A. |
| **Purpose** | This class is for creating events and retrieving event details. |
| **Collaborations** | This class is associated with both "User" and "Admin" classes. |
| **Attributes** | Capacity(Int), Date(varchar). |
| **Operations** | setDate(Date), returns null; getDate(), returns date; viewCapacity(), returns capacity; setCapacity(), returns null; |
| **Constraints** | N/A. |

# 9 Project Plan

Table 18: Project name time plan

| Id | Task | Start Date | Number of Days | Team Member |
|---|---|---|---|---|
| 1 | Requirements Gathering | 19/4/2021 | 7 | Everyone |
| 2 | Work on Wireframes | 25/4/2020 | 7 | Everyone |
| 3 | Work on GUI | 25/4/2020 | 15 | Mahi, Mariam |
| 4 | Work on Back-end | 4/05/2020 | 15 | Nour, Zeina |
| 5 | Work on SDD | 28/05/2021 | 7 | Everyone |
| 6 | Update SRS and SDD | 19/06/2021 | 7 | Everyone |

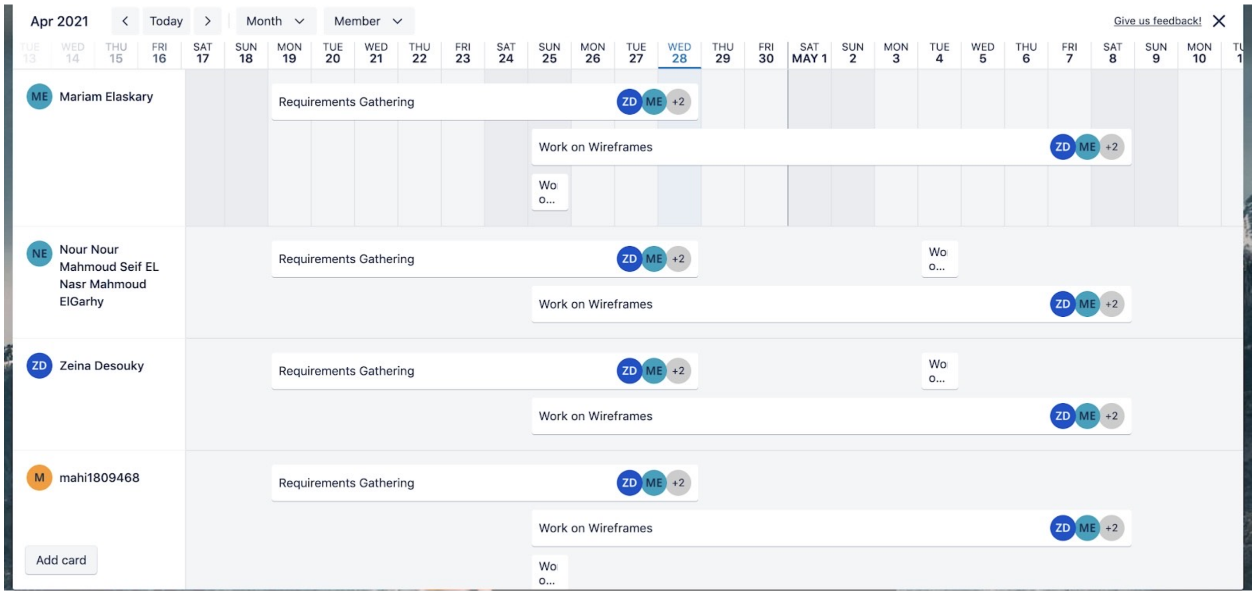


Figure 10: Project name GANTT Chart

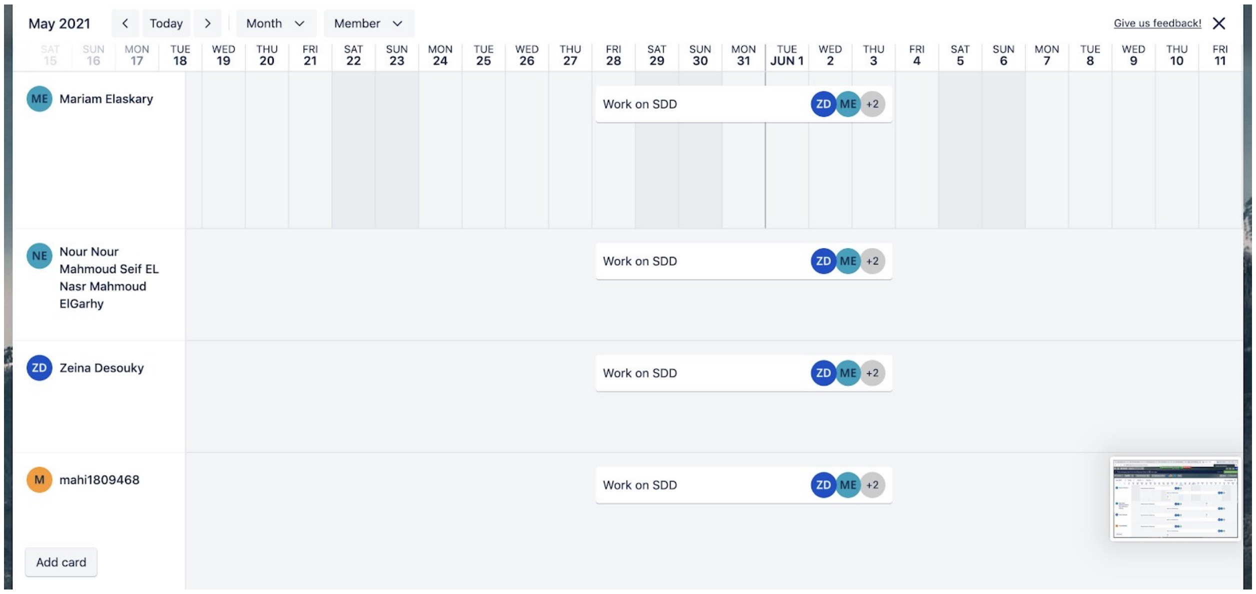


Figure 11: Project name GANTT Chart

## 10 Appendices

### 10.1 Definitions, Acronyms, Abbreviations

SQL: Structured Query Language. GUI: Graphical user interface.

### 10.2 Supportive Documents

*** Include in this section examples of any collected documents such as paper work samples, Communications with the client, etc.

## References

[1] RO Akinyede, TE Balogun, and Gabriel Iwasokun. “Design and implementation of an online booking system for a cinema house”. In: *Journal of Information and Computing Science* 12.2 (2017), pp. 113–122.