

# **Renting is easy for teachers (REFT)**

**Graduation Project**

**By**

**Osman Mamdouh Ramadan  
Maher Ahmed Ali Alshahbor  
Karim Allah Saad Elbanan  
Ahmed Mohamed Elmasarawy  
Mahmoud Refaat Osman  
Muaaz Reyad Hasan Abdelwahed  
Abdulrahman Ali Almowafy  
Muaaz Shaaban Abdelwahed**

**Supervised by**

**Prof. Dr. Wael Abdelkader    Dr. Ahmed Mohammed Rabie**

**2023/2024**

# Abstract

It is obvious that our country, Egypt, is adopting the new technologies to keep up with technological revolution in the world. We believe that modern technologies should be applied in all educational fields, because better education is central to human happiness and well-being. It also makes an important contribution to economic progress, as educated populations are Conscious, more productive, and earn more. So, our proposed system will target the education field. We are going to build a complete environment to meet the teacher's needs beginning from searching for the appropriate place for him, communicating with the owner of the place and completing the reservation process while ensuring security. The system has been developed using React, Node.js and a rental website. So, “Rental management system for teachers” is the title of our project and we call the system REFT.

**Keywords:** React; Node.js; Renting Website; Security; Teacher.

# **Acknowledgement**

We would like to thank our great parents who spare no effort for helping and supporting us all the time. And, we promise we will continue doing our best to make them proud of their sons/daughters. We, also, give many thanks to our supervisor, Assoc. Prof. Ahmed Mohamed rabie, who trusted and helped us and wish him all success in him life. And, for those students who will read this report in the next years, we wish it will help you in your projects. We had a great time in Faculty of Computers and Artificial Intelligence, Damietta University, we learned many things, and now it's time to use what we learned.

# Table of Content

<b>List of Figures.....</b>	<b>vi</b>
<b>List of Tables .....</b>	<b>viii</b>
<b>1 Chapter 1: Introduction.....</b>	<b>2</b>
1.1 Overview .....	2
1.2 Problem Definition (or Motivation) .....	2
1.3 Project Objectives .....	2
1.4 Project Scope.....	3
1.5 Project Timeline .....	4
1.6 Document Organization .....	5
<b>2 Chapter 2: Literature Review .....</b>	<b>7</b>
2.1 Background .....	7
2.2 Review of Relevant Work .....	9
2.3 Relationship between the Relevant Work and Our Own Work .....	14
2.4 Summary .....	16
<b>3 Chapter 3: System Analysis.....</b>	<b>18</b>
3.1 System Requirements .....	18
3.1.1 Functional Requirements .....	18
3.1.2 Non Functional Requirements .....	19
3.1.3 User Requirements .....	21
3.2 Development Methodology .....	22
3.2.1 Use Case Diagram.....	23
3.2.2 Activity Diagram .....	26

3.3	System Architecture .....	29
3.4	Tools and Languages.....	31
3.5	Summary .....	32
<b>4</b>	<b>Chapter 4: System Design.....</b>	<b>34</b>
4.1	Class Diagram .....	34
4.2	ERD Diagram.....	37
4.3	Interface Design .....	38
<b>5</b>	<b>Chapter 5: System Implementation.....</b>	<b>48</b>
5.1	Technologies .....	48
5.2	Sample Application Codes .....	51
5.3	System Testing .....	56
5.4	Results .....	58
5.5	Achieved Goals .....	58
<b>6</b>	<b>Chapter 6: Conclusion and Future Work .....</b>	<b>68</b>
6.1	Conclusion.....	69
6.2	Future Work .....	69
<b>7</b>	<b>References .....</b>	<b>70</b>
<b>8</b>	<b>Appendix A .....</b>	<b>72</b>

# List of Figures

Figure 2.1 Sign up page.....	10
Figure 2.2 Owner profile page.....	11
Figure 2.3 Property page.....	13
Figure 2.4 Profile page.....	14
Figure 2.5 Rentoaxis signup page.....	16
Figure 2.6 Booking the room.....	17
Figure 3.1 Use Case diagram .....	25
Figure 3.2 Activity diagram .....	28
Figure 4.1 class diagram .....	34
Figure 4.2 ERD diagram .....	37
Figure 4.3 Sign up page.....	43
Figure 4.4 Login page.....	44
Figure 4.5 Forget the password.....	45
Figure 4.6 Enter a new password.....	47
Figure 4.7.1 Home page.....	48
Figure 4.7.2 rest of Home page.....	49
Figure 4.8 Add hall page.....	48
Figure 4.9 Admin page.....	49
Figure 4.10 Hall details page.....	50
Figure 4.11 Dashboard.....	51
Figure 4.12 My places page.....	52

Figure 4.13 About us page.....	53
Figure 4.14 Contact us page.....	54
Figure 5.1 Code for login page.....	60
Figure 5.2 Code for signup page.....	61
Figure 5.3 Code for credentials check.....	61
Figure 5.4 Code for adding hall.....	62
Figure 5.5 Code for checking validity.....	63
Figure 5.6 Code for searching process.....	63

# List of Tables

Table 1.1 project timeline.....	4
Table 3.1 Non-functional requirements.....	24
Table 5.1 Black box testing.....	64
Table 5.2 White box testing.....	65
Table 5.3 Achieved goals.....	67



# **Chapter 1**

## **Introduction**

# **Chapter 1 Introduction**

## **1.1 Overview**

We are currently living in the era of technology, which plays a significant role in the progress of nations towards an era characterized by precision, efficiency, and saving time and effort in all areas of life, whether it is in engineering, medicine, agriculture, industry, or education. Our topic specifically focuses on the educational aspect because all other fields depend on it. It is the cornerstone or solid foundation upon which all other fields rely.

In this chapter, we will provide an introduction to our proposed project. First, in section 1.2 and section 1.3 we discuss the motivation and the objectives of our project respectively. Next, in section 1.4, we present the scope of the intended project. Then, Section 1.5 shows timeline of our project. Finally, section 1.6 provides a document organization including a brief description about the context of each chapter.

## **1.2 Problem Definition**

Egyptian families spend very large sums of money for private lessons for their children, even though government education is free and e-learning is available in an unprecedented manner. one of the most prominent reasons for the high cost of private lessons is that teachers rent places for large sums of money to give lessons to their students. We will create a website that allows the teacher to rent the place, but according to the number of hours he needs to teach. Consequently, the cost will be reduced for the teacher and thus for the Egyptian family, allowing them to spend the amount saved on other matters.

## **1.3 Project Objectives**

We will build a website that helps the teacher find the best place for him to give private lessons at a much lower price than the current situation and at the same time increase the profits

of the owner of the place. This is a list of the most prominent goals of the project.

- 1) It allows the owner of the place to share the location and photos and the cost per hour of the place
- 2) The place owner can determine the number of rooms to be rented and the appropriate timing for renting and even the appropriate days in the week
- 3) The teacher can search for available places near him
- 4) The teacher can rent a specific number of hours per month according to his personal needs
- 5) The place owner can determine the payment method, whether online or by physical methods

## **1.4 Project Scope**

The intended system aims to help the teacher find the appropriate place for him to give educational lessons by controlling the area of the place, the capacity of students, the cost of rent, and the appropriate location. In addition, it allows the owner to publish his property on our website with the needed information, such as the location, pictures of the place, the rental price, and the available days and hours without further trouble on both sides to find the appropriate option for them.

## 1.5 Project Timeline

#	Task Name	Start	Finish
1	Background research	20-10-2023	30-10-2023
2	Determine objectives	01-11-2023	20-11-2023
3	Review related work and papers	21-11-2023	10-12-2023
4	Determine needed components	11-12-2023	22-12-2023
5	Design diagrams	26-01-2024	07-02-2024
6	Presentation for phase 1	26-01-2024	07-02-2024
7	Implementation	12-2-2024	15-5-2024
8.1	Front-end development	12-2-2024	1-3-2024
8.2	Back-end development	2-3-2024	5-4-2024
8.2.1	Database design	15-4-2024	15-5-2024
9	Testing	15-6-2024	25-6-2024
9.1	Integration Testing for the system	18-6-2024	21-6-2024

Table 1.1 Project Timeline

## 1.6 Document Organization

This document consists of six chapters in addition to one appendix. A brief description about the contents of each chapter is given in the following paragraphs:

**Chapter 2:-** Literature Review, provides the reader with an overview of the previous related work, common technologies used, and the relation between our work and the relevant work.

**Chapter 3:-** System Analysis, includes the analysis of existing system, system requirements, use requirements, system architecture, development methodology languages used in our system.

**Chapter 4:-** System Design, provides the system design including class diagram. design and interface design.

**Chapter 5:-** System implementation, shows the process of mapping design into implementation, sample application code, system testing, results of the investigation, and goals achieved.

# **Chapter 2**

## **Literature Review**

# Chapter 2: Literature Review

In this chapter, we provide a background of the tools and techniques needed to build our system. We also review the previous related work to our system and the common technologies that are used. At the end of the chapter, we provide a comparison between the system we are going to build and the related systems.

## 2.1 Background

The system we are going to build will depend on Google Map API and web application to operate. The following subsections provide a brief background information about tools, and techniques needed to build our system including programming languages and frameworks. There is a total of two major parts of the project plan, which are the frontend and backend parts. In this scientific research project HTML, CSS, Tailwind CSS, JavaScript, and React are used for the frontend design of the system. MongoDB and Express are used for creating and designing the database and Backend part. For writing all the codes Visual Studio Code editor is used.

### HTML

HTML stands for Hypertext Markup Language which allows the user to make and structure sections, paragraphs, headings, titles, line breaks, add media, links, blockquotes, etc. for websites and applications.

### CSS

CSS stands for Cascading Style Sheets which is a simple design language intended to simplify the method of constructing this website presentable. It's designed to enable the separation of presentation and content, including layout, colors, spacing, padding, fonts, and so on.

## **Java Script**

JavaScript is a scripting or dynamic computer programming language that allows the implementation of complex features on web pages and client-side scripts to interact with the user and make dynamic web pages.

## **React Bootstrap [1]**

React Bootstrap is an open-source library that combines the flexibility of React with the styling capabilities of Bootstrap. It offers a wide range of UI components, such as buttons, modals, navigation bars, forms, and more. These components are designed to be easily customizable and reusable across different projects.

## **React [2]**

React is the library for web and native user interfaces. Build user interfaces out of individual pieces called components written in JavaScript.

## **Node.js [3]**

Node.js is a runtime environment that allows developers to run JavaScript on the server side. Built on Chrome's V8 JavaScript engine, it enables the development of scalable and high-performance network applications.

## **Express.js [4]**

Express.js is a fast and minimalist web application framework for Node.js. It provides a simple and flexible way to build web applications and APIs. With Express.js, you can easily handle HTTP requests, define routes, and implement middleware for tasks like authentication and error handling. It has a robust ecosystem of plugins and middleware that can be used to extend its functionality.



## PostgreSQL [5]

PostgreSQL is an advanced relational database system. PostgreSQL supports both relational (SQL) and non-relational (JSON) queries. PostgreSQL is free and open-source.

## 2.2 Review of Relevant Work

The following are some of the research and papers that study Renting or reserving places for various purposes:

- Dipta Voumick, Prince Deb, Sourav Sutradhar, Mohammad Monirujjaman Khan [6] An Online Based Smart House Renting Web Application. In this paper, the system can provide a framework that allows managers to conduct reasonable transactions within a limited time frame. This system is meant to satisfy the needs of rental house owners. Several user-friendly interfaces have also been adopted. The real time chat system will bring fluidity over the usage and connection between owner and tenants. Also, the location tracing system will be a major advantage for users as it will be easy to find the location of the house on a map. In addition, for the concern of security, this system has optimized secure and private data storage and verification system.
- Landlords' powers in the application:
  - Listing properties with details and photos
  - Managing rental agreements and leases electronically
  - Processing rent payments online
- Tenants' powers in the application:
  - Searching for rental properties based on their preferences
  - Applying for rentals online
  - Viewing and signing rental agreements electronically

**Sign Up**  
Please fill this form to Create an Account.

**RegisterID**

**Name**

**Username**

**Email**

**Contact**

**Password**

**Confirm Password**

Already have an account? [Login here.](#)

Figure 2.1 sign up page

# Owner Profile

Enter Your Name

Enter ID

Enter Address

Enter Your Mobile No

Enter Your Email

Enter Your House-ID

[Submit](#) [Reset](#) [Home](#)

Figure 2.2 owner profile page

-Mandeep Katre (Professor), Kumar Abhay Gupta, Shipra Katiyar, Saumya Shahi, Shreya Awasthi. A Review Paper on PG Recommendation System. Rental Zone [7]

In this paper rental website development is discussed. Through this website, tenants can rent properties and landlords can upload their property details for rent. This website will help users to give or take rent houses without dealing with flat brokers to finally reach an agreement that suits the interests of all parties. The packaging and shifting services facility will help users shift valuable things safely to their destinations.

The mechanism of action of the application is:

1. Search by preferred location, currency, distance, etc.
2. We provide all hostels, PG, Rental rooms, etc.
3. Students can hire a teacher close to their studies.
4. Daily, weekly, and monthly visit justice.
5. Provides quality Tiffin daily, weekly, and monthly service.

#### A. Admin Module

- 1) Dashboard: In this section, the administrator can see all the details briefly like country total, city total, total owner, and PG theme.
- 2) Region / City: In this section, the administrator can manage status (add / update).
- 3) Record Owner: In this section, the administrator can view and edit the registered owner.

#### B. PG Owner Module

- 1) Dashboard: In this section, the owner can view all the details briefly like pg. listed list, complete booking, new booking, total guaranteed booking, and total canceled booking.
- 2) List Your PG: In this section, the owner can list their page.
- 3) Received Request Booking: At this stage, the administrator may view the new booking and is entitled to a guaranteed booking.

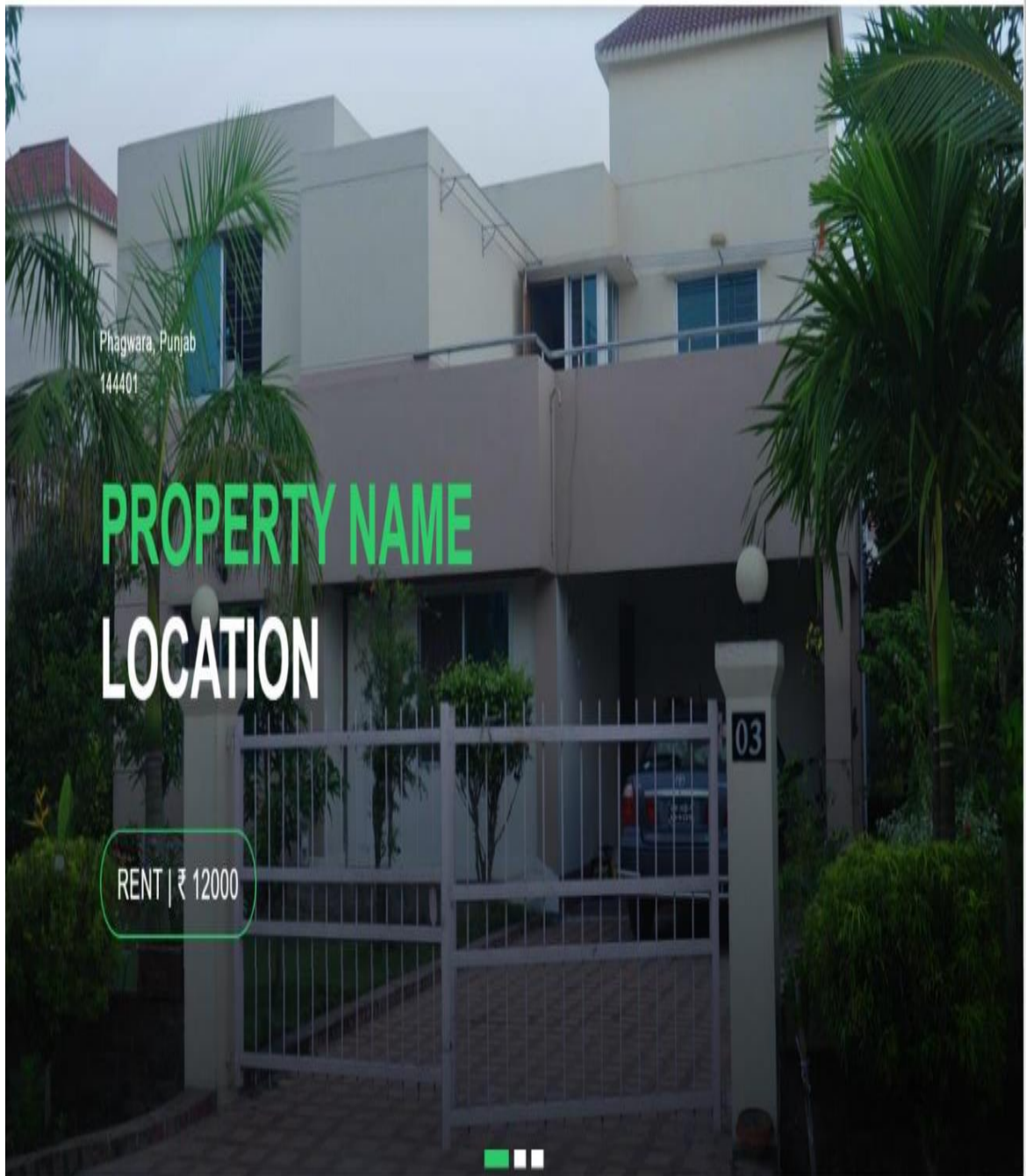


Figure 2.3 property page

**THE RENTAL ZONE**

## Login

Login as: Owner ▾

Owner  
Renter

Enter Your email

---

Enter Your Password

---

[Forgot password?](#)

[Login](#)

Don't have an account? [Register here](#)

Figure 2.4 Profile page

-Murahari Prithvi Yash, Chinmay Choudhary, Akanksha Lakra, Swati Dewangan. RentoAxis: Android App for Paying Guest Management. [8]

This paper presents the design and implementation of an Android-based Paying Guest (PG) management app named RentoAxis. The proposed system comprises an android application where (i) the PG owners can advertise the PG details by registering into the application. (ii) the user (common people) can search for the PG, apartments, and houses all over Durg and Raipur. (iii) the customers can also search the PG information based on some parameters such as nearby colleges, markets, etc. In other words, this application will help PG owners improve the management of their community and the tasks related to it.

A. Administrator Module:

- Manages all the information and has access rights.
- Check the details of the PG owner as well as the customer.
- View/Delete the details related to PG, owner, and user.

B. Owner Module:

- The PG Owner will follow the registration procedure by using identity proofs.
- They can easily add details of the Apartment, Rooms, and Paying Guests and can update or delete the details according to the need.

C. User Module:

- The user will also register by their unique ID.
- Users can search for hostel rooms, paying guests, etc., and get the room details, room rent, address of the room, pictures of the room, etc. across the city.

This app works in three phases:

- Phase 1- RentoAxis: Login Screen



- Phase 2- RentoAxis: PG list and Filter Option
- Phase 3- Accommodation Details and Reservation

Stages of application:



Figure 2.5 Rentoaxis sign up page

The Snapshot (a) shows the users can sign up or log in to link their data to this email and facilitate the browsing process in the application.





Figure 2.6 Booking the room

The snapshot (b) shows the client booking the room in which he will stay after settling.

## **2.3 Relationship between the Relevant Work and Our Work**

The previously mentioned related work shows that designed systems that study the Rental Management Systems and Paying Guest Management but have some drawbacks we will overcome them with the following:

1. Fills a specific market need: Our project addresses the unique needs of teachers seeking affordable lesson spaces, which is not addressed by other platforms.
2. Reduced cost for teachers: By connecting teachers directly with space owners, our project eliminates the need for middlemen and reduces the cost of renting spaces.
3. Increased profits for space owners: Our project allow space owners to set their prices and reach a wider audience of potential renters.
4. Secure and convenient platform: Our project provides a secure platform for booking and managing rentals, as well as a system for reviews and ratings.
5. Limited focus: These platforms are not designed specifically for private lessons, leading to a cluttered search experience for teachers and potentially unsuitable spaces.

6. High cost: Renting spaces through other applications can be expensive, particularly for teachers who need them regularly.
7. Lack of specific features: These platforms lack features tailored to private lessons, such as booking specific hours, managing student schedules, or providing feedback.
8. Diversification of income streams: Renting spaces for private lessons can be an additional income stream for space owners, alongside any existing rental activities.
9. Reduced risk of cancellations: A secure booking system and clear cancellation policies can help teachers avoid lost income due to last-minute cancellations.

## 2.4 Summary

The previous sections show the needed background information to understand how we are going to build a system that can provide places for private lessons while saving the time and effort of teachers in searching for specific locations to teach the subjects. Section 2.1 provides information on software tools, and techniques needed to develop the web application and how to employ the technologies to suit the functions of the application. Section 2.2 discusses relevant work based on research and papers that study Rental Management Systems and Paying Guest Management. The first app intended to provide landlord/agent/tenant/prospective tenant with information on house records, second app is a website that will help users to give or take rent houses without dealing with flat brokers to finally reach an agreement that suits the interests of all parties, last app is an application for helping PG owners to improve the management of their community and the tasks related to it. In section 2.3, we discuss the relationship between our intended project and the related work. We seek to build a platform that facilitates affordable private lessons while increasing profits for space owners., Also, provides a secure platform for booking and managing rentals, as well as a system for reviews and ratings.

# **Chapter 3**

## **System Analysis**

# Chapter 3: System Analysis

In this chapter we provide a detailed knowledge about our system including functional requirements, non-functional requirements, user requirements, system architecture, use case and sequence diagrams. We can divide the process of analysis into three parts, which are determining requirements, determining system architecture, and determining development methodology. Section 3.2, Section 3.3, and Section 3.4 will discuss those three parts. In the end of this chapter, we provide information about the tools and languages we needed to use to build our system.

## 3.1 system requirements

System requirements are the needed configurations for the system to operate efficiently. The next three subsections will discuss the functional requirements, Nonfunctional requirements, and user requirements.

### 3.1.1 Functional requirements

In systems engineering, functional requirements are directly concerned with the system services, where a function is described as a specification of behavior. In this subsection, we list the functions required in our system. We provide a description for each function [12]

#### **Functional requirements:**

1. It allows the owner of the place to share the location and photos of the place

The owner of the place can share the geographical location of the place, upload pictures of the rooms he wants to rent, and also determine the appropriate financial cost for one hour of rent.

2. The place owner can specify the number of rooms and appropriate timings

The owner of the place determines the number of rooms he wants to share on the site and also determines the number of hours during which he wants the rent to be made.

3. Allows the teacher to search for nearby places

The site helps the teacher by showing him the nearest geographical places

The teacher can also search for available places and hours in any city he wants to search for.

4. The teacher can rent a specific number of hours according to his need

The teacher can rent as many hours as he wants on the days of the week he wants as long as it is available on the website.

5. Online payment available

The teacher can choose the payment method, either online or in the traditional way.

6. Add some laws and conditions

The owner of the place can set some conditions in addition to the general site rules, and the teacher must adhere to those conditions, and if he violates them, he is subject to fines.

### **3.1.2 Non-functional requirements**

The system needs to operate efficiently and meet the requirements. Any failure of the components of the systems may lead to one or more of functions stopping or being misused. A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system. [12]

## Non-functional requirements

Table 3.1 Non-functional requirements

#	Non-functional requirements	
1	availability	ensures that the app will work stably for a certain period, e.g. rare downtimes throughout the year 24/7
2	performance	assesses how fast the app is;
3	capacity	assesses the amount of data or services the app can handle.
4	recoverability	ensures that the app can recover all the data after the system failure or restore the system to certain parameters
5	reliability	defines that the app will work in a certain environment or for a specific period of time without any failures
6	scalability	determines that the app will continue functioning properly after its size or volume changes
7	usability	defines how easy a user can interact with the app's interface, e.g. the screen color, buttons size, etc.



## **Non-functional requirements**

### **Availability**

Easy design that is suited for different communities to use.

### **Performance**

The site is fast in using

### **Capacity**

Low storage space consumption.

### **recoverability**

System response within a few seconds.

### **Reliability**

System should be safe so patient can rely on

### **Scalability**

The site can work with many locations in many different cities

### **usability**

Easy to use and understand for different users

## **3.1.3 User requirements:**

The user of this site is in fact two users, the owner of the place (the lessor) who wants to rent one or more rooms to teachers according to the time that suits him, and the other user is the teacher who is looking for a suitable place to reserve for a number of hours per month at a price suitable for him.

### **Place owner requirements**

1. Login to the system
2. Share the geographical location of the place
3. Determine the number of rooms
4. Determine appropriate timings
5. Determine the payment method
6. Add an electronic wallet number
7. Determine the money per one hour
8. Establish special conditions for it
9. Delete the place

### **The Teacher Requirements**

1. login to the system
2. search for the nearest places location
3. Search for places in other cities
4. Choose a suitable place for him
5. Choose the appropriate room for the number of students he has
6. Choose the number of hours he wants
7. Choose a payment method
8. Cancel reservation

## **3.2 Development Methodology**

After we knew the basic structure of the system. We are going to view all of its functions, the relation between them, and the Activity diagram in the following subsections.

### 3.2.1 Use Case Diagrams

A use case diagram is a visual representation of the interactions between users (actors) and a system, illustrating the different ways the system can be used to achieve various goals. It is a fundamental tool in software engineering, particularly in the early stages of system design, to capture functional requirements. The diagram helps stakeholders understand the system's functionality and how different users will interact with it. [13]

Actors in a use case diagram represent roles played by users or other systems that interact with the system being designed. These actors can be human users, external systems, or hardware devices. Use cases describe specific functionalities or services provided by the system, depicted as ovals within the diagram.

The primary purpose of a use case diagram is to identify and clarify the requirements of a system. It provides a clear and concise way to document the interactions and can help ensure that all user needs are considered during the development process. Use case diagrams also facilitate communication between developers, designers, and stakeholders by providing a common language and visual context.

Each use case should be meaningful to the actors and should result in some observable value or change. Use case diagrams typically include associations, which are lines connecting actors to use cases, indicating the interaction. Relationships such as "include" and "extend" can also be shown, highlighting dependencies or optional functionalities.

"Inclusion" relationships are used to show that a use case always uses the behavior of another use case, making the interaction modular and reusable. "Extension" relationships

indicate optional or conditional behavior, allowing for flexibility and variability in how use cases are executed.

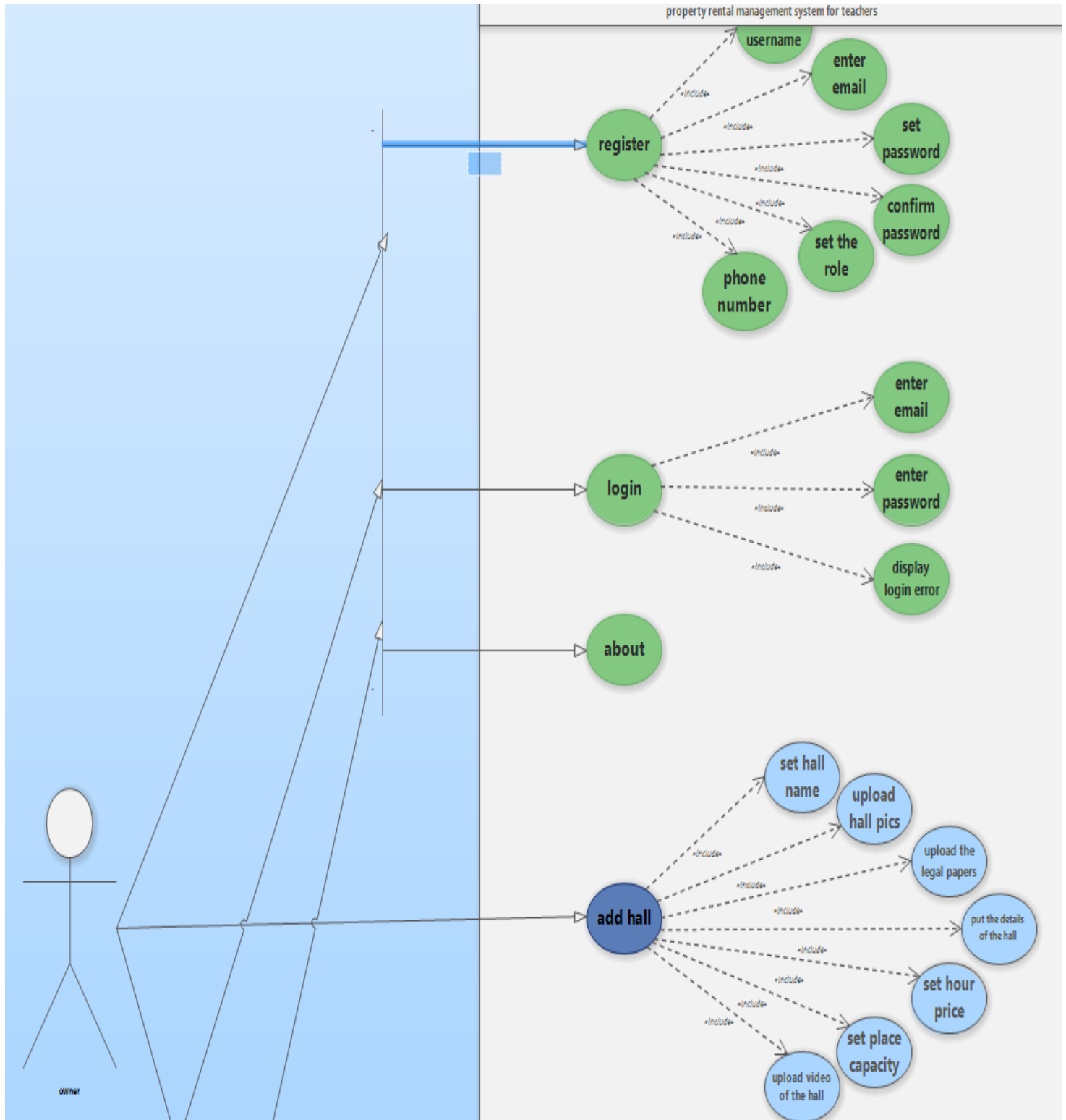
Use case diagrams do not specify the internal workings of the system but focus on what the system should do. They are often accompanied by detailed textual descriptions of each use case, providing further insight into the steps involved and the expected outcomes.

By visualizing the use cases, designers can better identify redundancies, gaps, or unnecessary complexities in the system. This can lead to more efficient and user-friendly designs. Use case diagrams can also aid in test case development by providing scenarios that need to be tested to ensure the system meets user requirements.

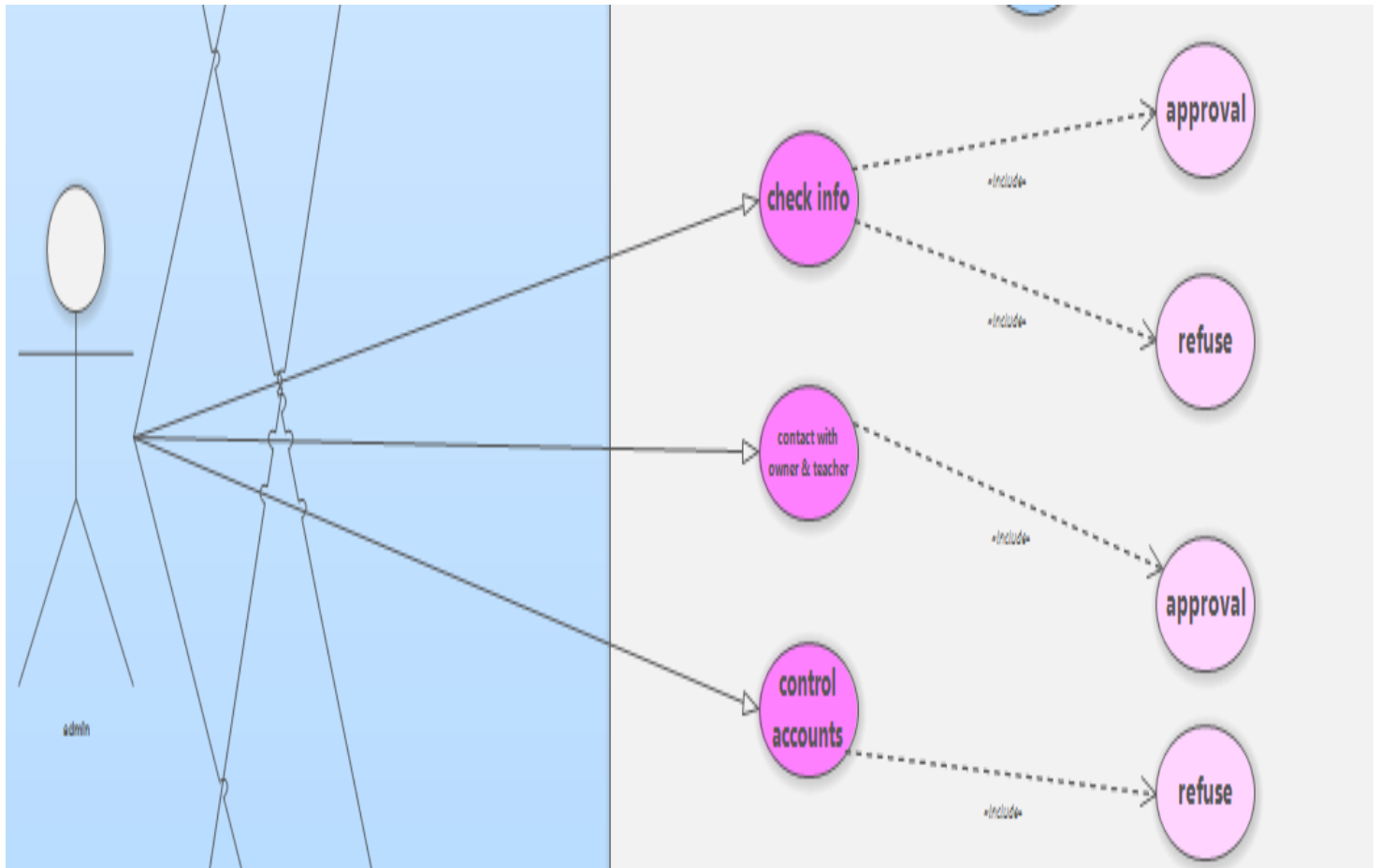
They are typically created during the requirements gathering phase but can be updated throughout the project lifecycle as requirements evolve. Tools like UML (Unified Modeling Language) are commonly used to create use case diagrams, offering standardized notation and symbols.

In summary, a use case diagram is a crucial tool for capturing functional requirements, facilitating communication among stakeholders, and guiding the design and testing of a system. It ensures that all user interactions are considered, leading to a more complete and well-designed system.

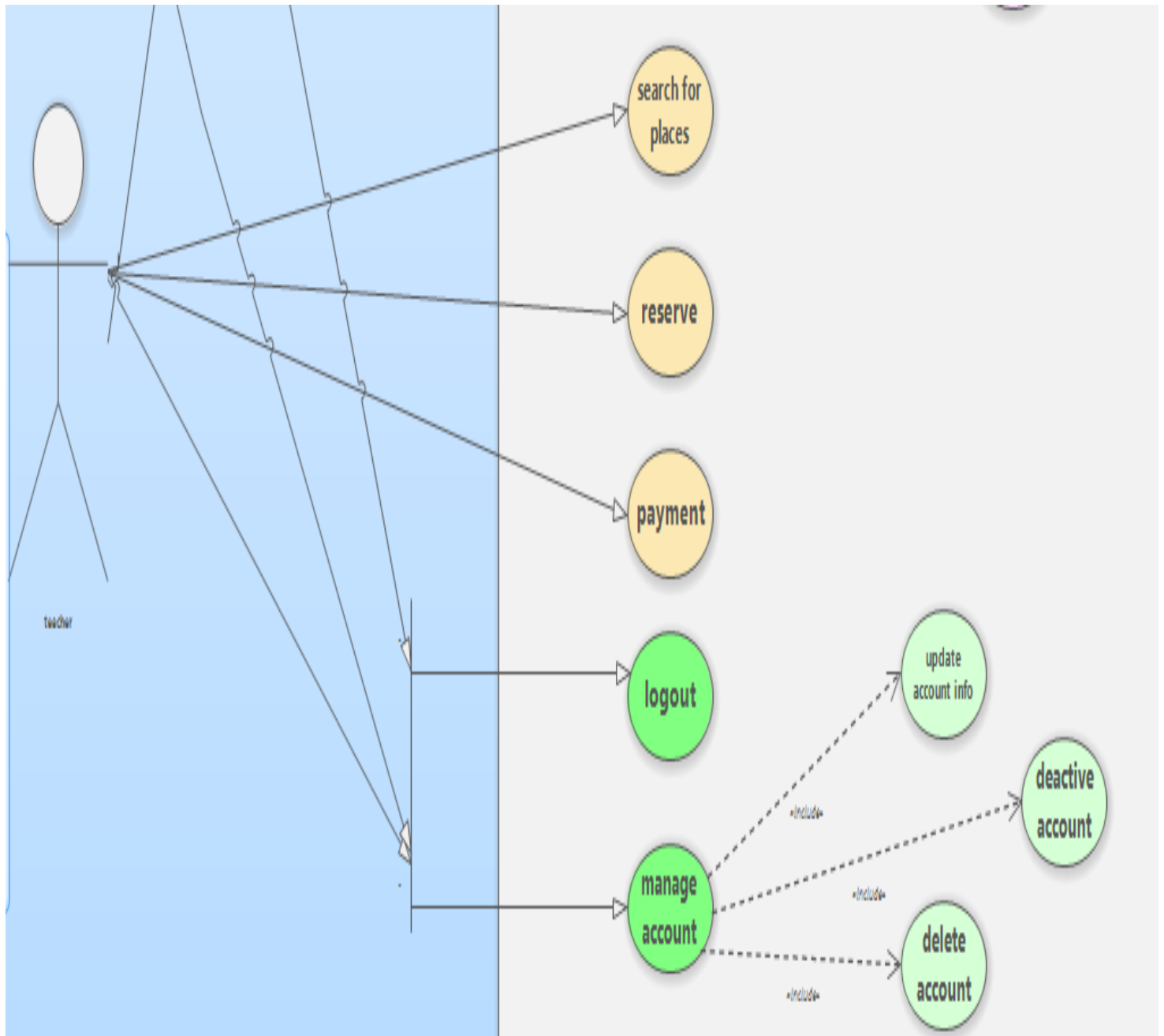
First, with a use case diagram, we will specify the expected behavior (what) of the system, not the exact method of making it happen (how). This helps us to design the system from the end user's perspective. Figure 3.3 shows the use case diagram where the system boundary is the EduNet and the actor is the place owner and the teacher .



**Figure 3.1.1 Owner use case diagram**



**3.1.2 admin use case diagram**



**3.1.3 tenant use case diagram**

### **3.2.2 Activity Diagram**

A class diagram is a type of static structure diagram in the Unified Modeling Language (UML) that describes the structure of a system by showing its classes, their attributes, operations (methods), and the relationships among objects. It is one of the most common types of diagrams used in software engineering to model the static structure of a system. The main components of a class diagram are classes, attributes, operations, and relationships.

Classes are depicted as rectangles divided into three compartments: the top compartment contains the class name, the middle compartment contains the attributes, and the bottom compartment contains the methods. Attributes represent the properties or data held by the class, while operations represent the behaviors or functions the class can perform.

Relationships between classes include associations, dependencies, generalizations, and realizations. Associations represent the connections between classes, indicating how objects of one class interact with objects of another. Multiplicity indicators (e.g., 1, 0.., 1..) on associations specify the number of instances involved in the relationship.

Inheritance (or generalization) relationships depict a hierarchy between a more general superclass and more specific subclasses. This allows subclasses to inherit attributes and methods from the superclass. Aggregation and composition are special types of associations that represent whole-part relationships, with composition implying a stronger ownership.



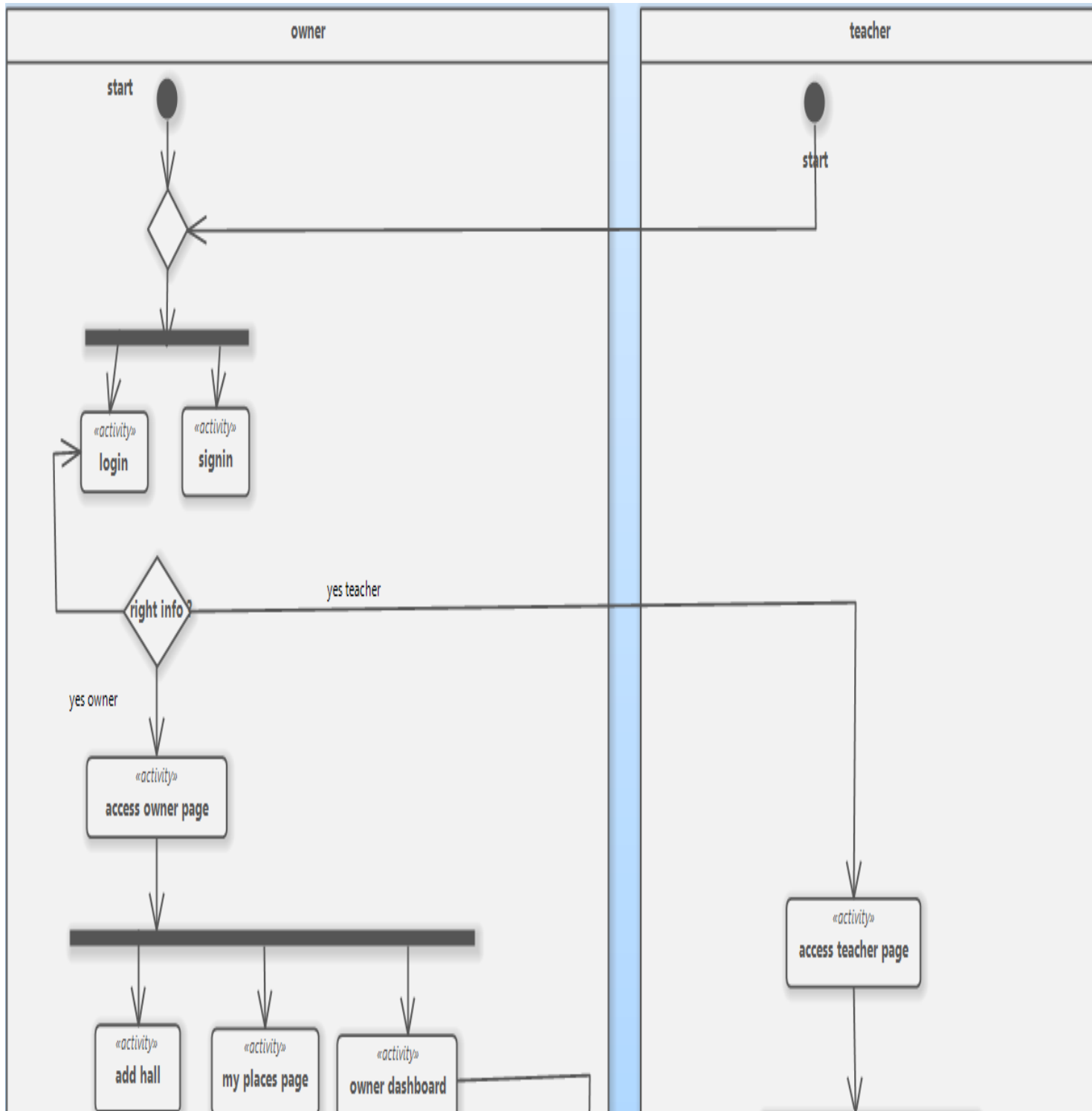
Class diagrams help in understanding and documenting the system architecture, and they are essential for both object-oriented analysis and design. They serve as a blueprint for constructing the software system, guiding developers in implementing the system's classes and relationships.

In addition to modeling the structure, class diagrams can also represent abstract classes, interfaces, and enumerations. Abstract classes are depicted in italics and cannot be instantiated, while interfaces define methods without implementation, which implementing classes must provide.

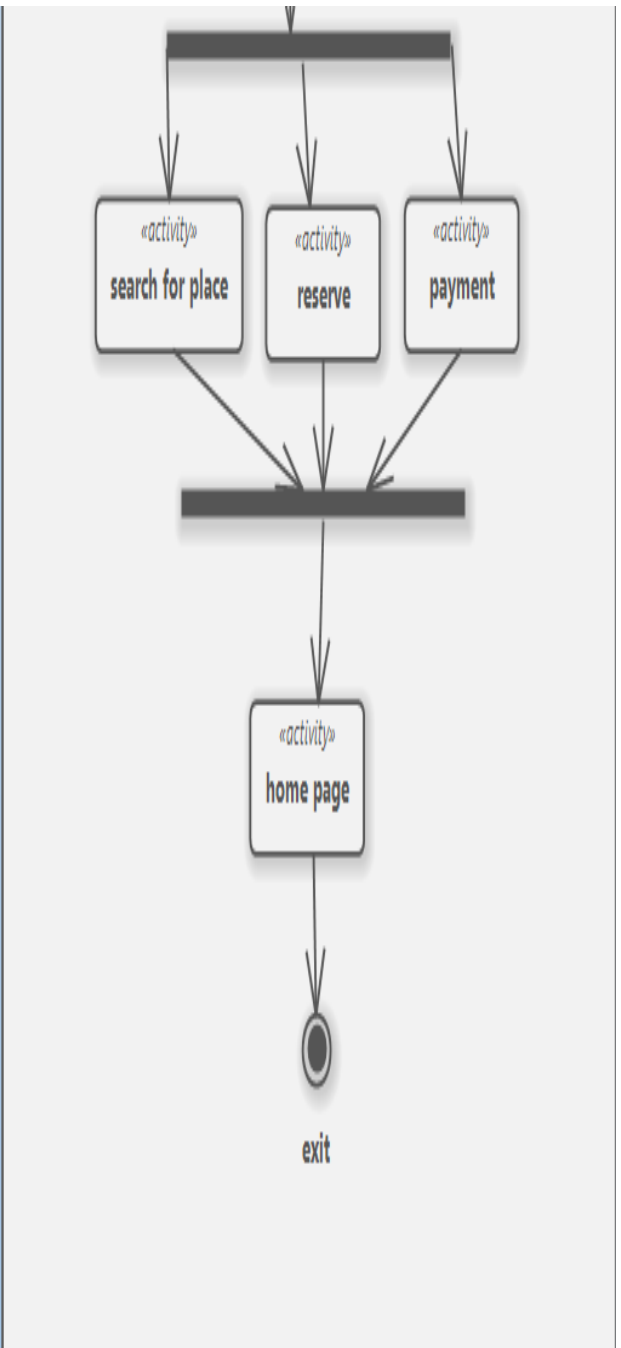
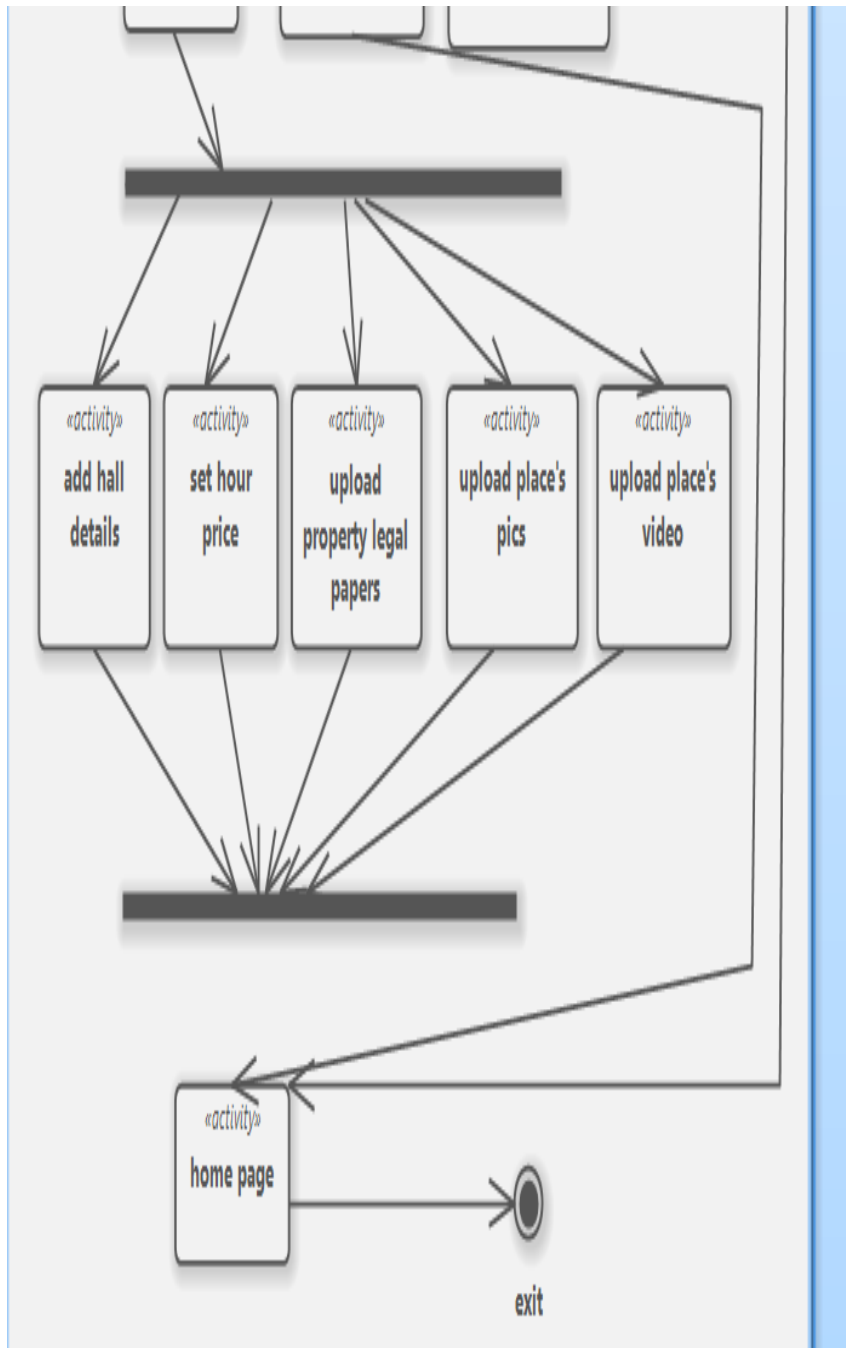
Class diagrams can evolve over time, from high-level conceptual models to detailed design models as the system's requirements and design become more concrete. They play a crucial role in object-oriented programming languages like Java, C++, and Python, where classes and objects are fundamental constructs.

## **Activity Diagram**

We use Activity Diagrams to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on the condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram. UML models basically three types of diagrams, namely, structure diagrams, interaction diagrams, and behavior diagrams. An activity diagram is a behavioral diagram i.e. it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. We can depict both sequential processing and concurrent processing of activities using an activity diagram. They are used in business and process modelling where their primary use is to depict the dynamic aspects of a system. An activity diagram is very similar to a flowchart. So let us understand if an activity diagrams or flowcharts are any different :



**3.2.1 stage 1 activity diagram**



3.2.2 stage 2 activity diagram

## 3.2 System architecture

### 1. Frontend:

- User Interface (UI): The frontend should have a user-friendly interface for both property owners and teachers.
- Search and Filters: Implement a robust search functionality with filters like location, size, amenities, etc.
- User Authentication: Secure registration and login system for teachers and property owners.
- Property Listings: Display property listings with details, pricing, and availability.
- Booking System: Enable teachers to submit booking requests, and property owners to manage and confirm bookings.

## **2. Backend:**

- **Server:** Host your backend on a server. Popular choices include AWS, Google Cloud, or Azure.
- **Backend Framework:** Use a backend framework like Django (Python), Ruby on Rails (Ruby), or Express.js (Node.js).
- **Database:** Store property information, user details, and bookings in a relational database (e.g., PostgreSQL, MySQL).
- **Authentication:** Implement a secure authentication system for user login and authorization.
- **APIs:** Create APIs to handle communication between the frontend and backend.

## **3. User Management:**

- **User Profiles:** Allow users to create and manage their profiles.
- **Messaging System:** Implement a messaging system for communication between teachers and property owners.

## **4. Listing Management:**

- **Property Upload:** Create a feature for property owners to upload details, images, and availability of their places.
- **Content Management System (CMS):** Implement a CMS for property owners to manage their listings easily.

## **5. Booking System:**

- Reservation Handling: Develop a system for handling booking requests and confirmations.
- Calendar Integration: Integrate a calendar system to show property availability.

## **6. Payments:**

- Payment Gateway: Integrate a secure payment gateway (e.g., Stripe, PayPal) for handling transactions.
- Pricing System: Implement a pricing system based on property features, location, and duration of stay.

## **7. Notifications:**

- Email Notifications: Send email notifications for account verification, booking confirmations, and other relevant updates.
- In-App Notifications: Implement in-app notifications for real-time updates.

## **8. Security:**

- SSL Certification: Ensure secure data transmission with SSL.
- Data Encryption: Encrypt sensitive data, especially user credentials and payment information.

## **9. Analytics and Monitoring:**

- Analytics Tools: Integrate tools like Google Analytics to track user behavior.
- Error Monitoring: Implement a system to monitor and log errors for quick troubleshooting.

## 10. Scaling:

- **Scalable Infrastructure:** Design the architecture to scale horizontally as the user base grows.

Remember, this is a high-level overview, and the specific technologies you choose will depend on your preferences, expertise, and other project requirements. Additionally, consider legal aspects, such as terms of service and privacy policies, to protect both property owners and teachers using your platform.

## 3.3 Tools and Languages

Developing our software application can be divided into main two parts, which are the design part and the implementation part. The design part involves designing diagrams and designing user interface of the site .The implementation part involves programming languages, IDEs, frameworks, and libraries. The following list shows the needed tools for the software development and a brief description about their usages:

1. **Software Ideas Modeler** – it is used to draw the UML diagrams.
2. **Figma** – it is used to design user interfaces and prototypes.
3. **Visual studio code** – it's used to write the programming codes



## 3.4 Summary

In this chapter we provide the reader with detailed knowledge about our system. Section 3.1 provides system requirements Which is divided into functional and non-functional requirements, and user requirements which specify some different specifications for users, Section 3.2 includes system architecture which describe its major components, their relationships, and how they interact with each other, Section 3.3 provides development methodology which includes UML diagrams that shows the details of how the system will function. In the end of the chapter we listed the needed tools for us to build the system.

# **Chapter 4**

## **System Design**

# Chapter 4: System Design

## 4.1 class diagram

Class diagrams are the main building blocks of every object-oriented method. The class diagram can be used to show the classes, relationships, interface, association, and collaboration. UML is standardized in class diagrams. Since classes are the building block of an application that is based on OOPs, so as the class diagram has an appropriate structure to represent the classes, inheritance, relationships, and everything that OOPs have in their context. It describes various kinds of objects and the static relationship between them.

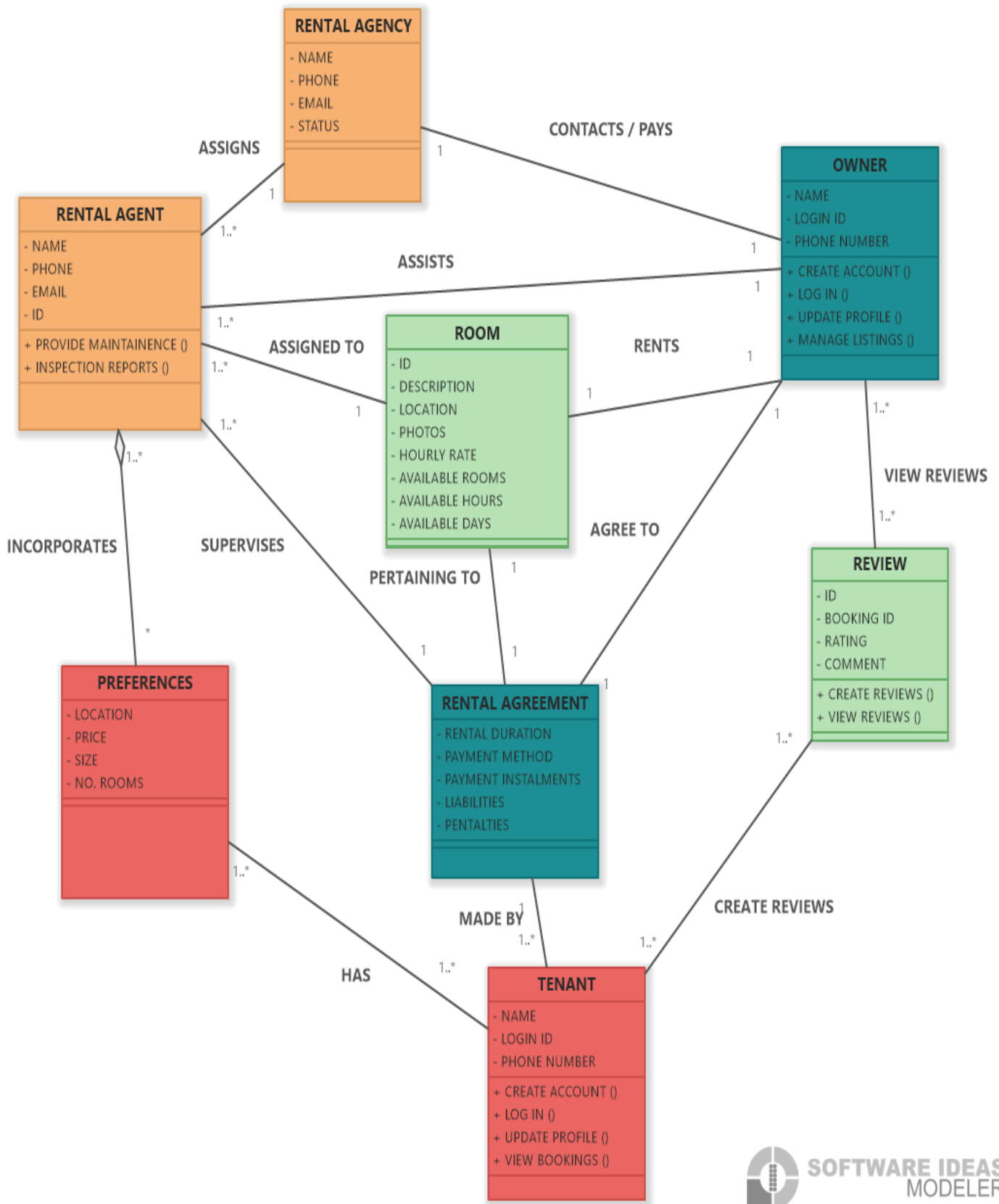


Figure 4.1 class diagram

## 4.2 ERD diagram

An Entity-Relationship Diagram (ERD) is a type of flowchart that illustrates how entities in a system relate to one another. ERDs are a crucial part of database design, helping to map out the structure and relationships within a database. They visually represent data and their interconnections using entities, attributes, and relationships.

**\*Entities\*** are objects or concepts, often represented by rectangles, that store data within the database. Common examples include "Customer," "Order," and "Product." Each entity has **\*attributes\***, depicted as ovals, which are the specific pieces of information stored about the entity, like "Customer Name" or "Order Date."

**\*Relationships\*** show how entities interact with each other, represented by diamonds or lines connecting entities. These can be one-to-one, one-to-many, or many-to-many, describing the nature of the interactions between entities. For instance, a "Customer" can place "Orders," establishing a one-to-many relationship.

ERDs also include **\*primary keys\***, unique identifiers for entities, and **\*\*foreign keys\***, which link entities together. Primary keys are usually underlined in the diagram. **\*\*Cardinality\*** and **\*participation constraints\*** specify the numerical limitations and obligatory participation of relationships between entities.

For example, a one-to-many relationship between "Customer" and "Order" signifies that one customer can place multiple orders, but each order is placed by only one customer.

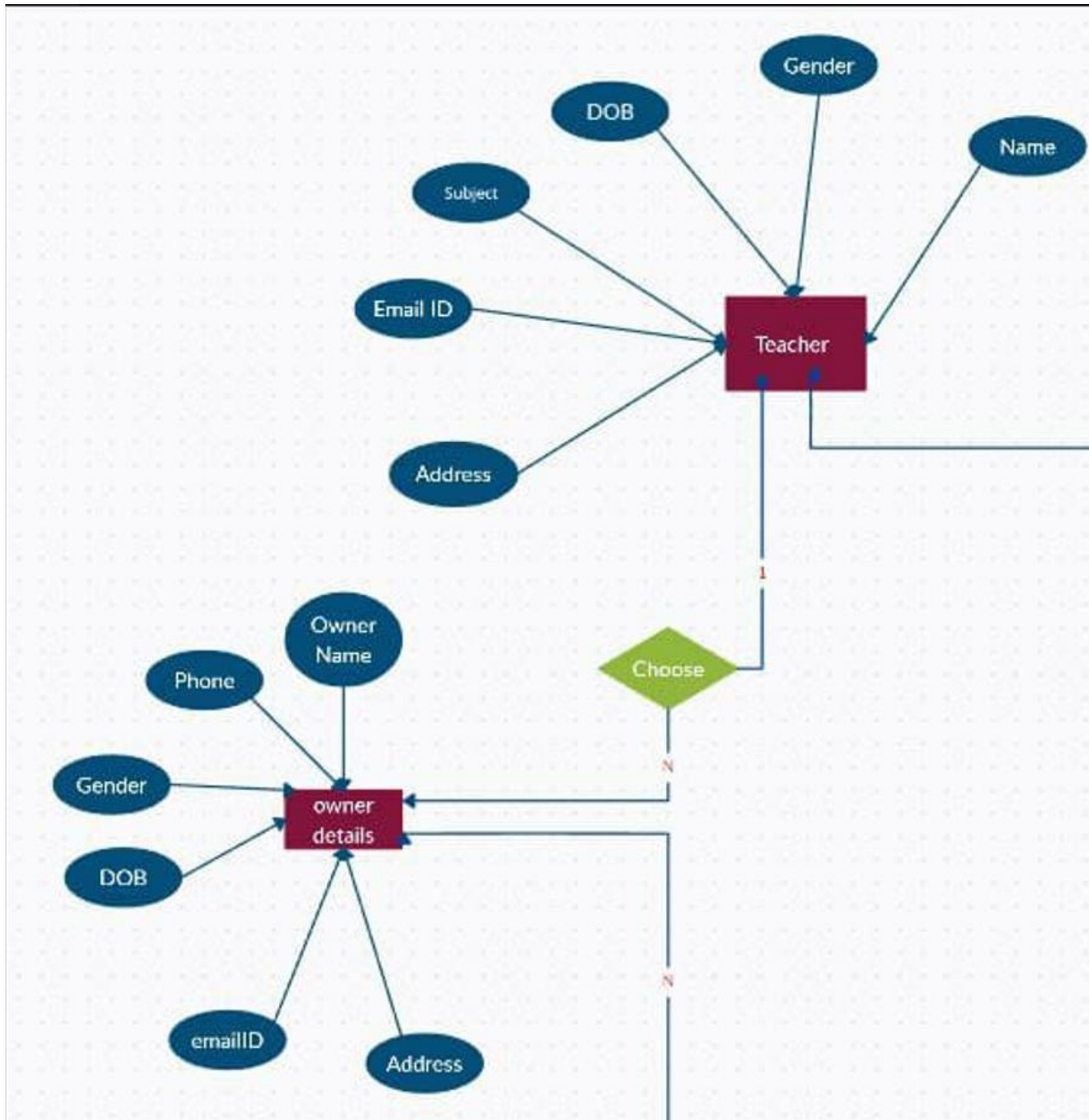
**\*Attributes\*** can be simple or composite (made up of multiple attributes), single-valued or multi-valued, and stored or derived.

\*Strong entities\* exist independently within the database, while \*weak entities\* depend on strong entities for their identification. \*Associative entities\* connect entities with a many-to-many relationship, adding extra attributes to the relationship.

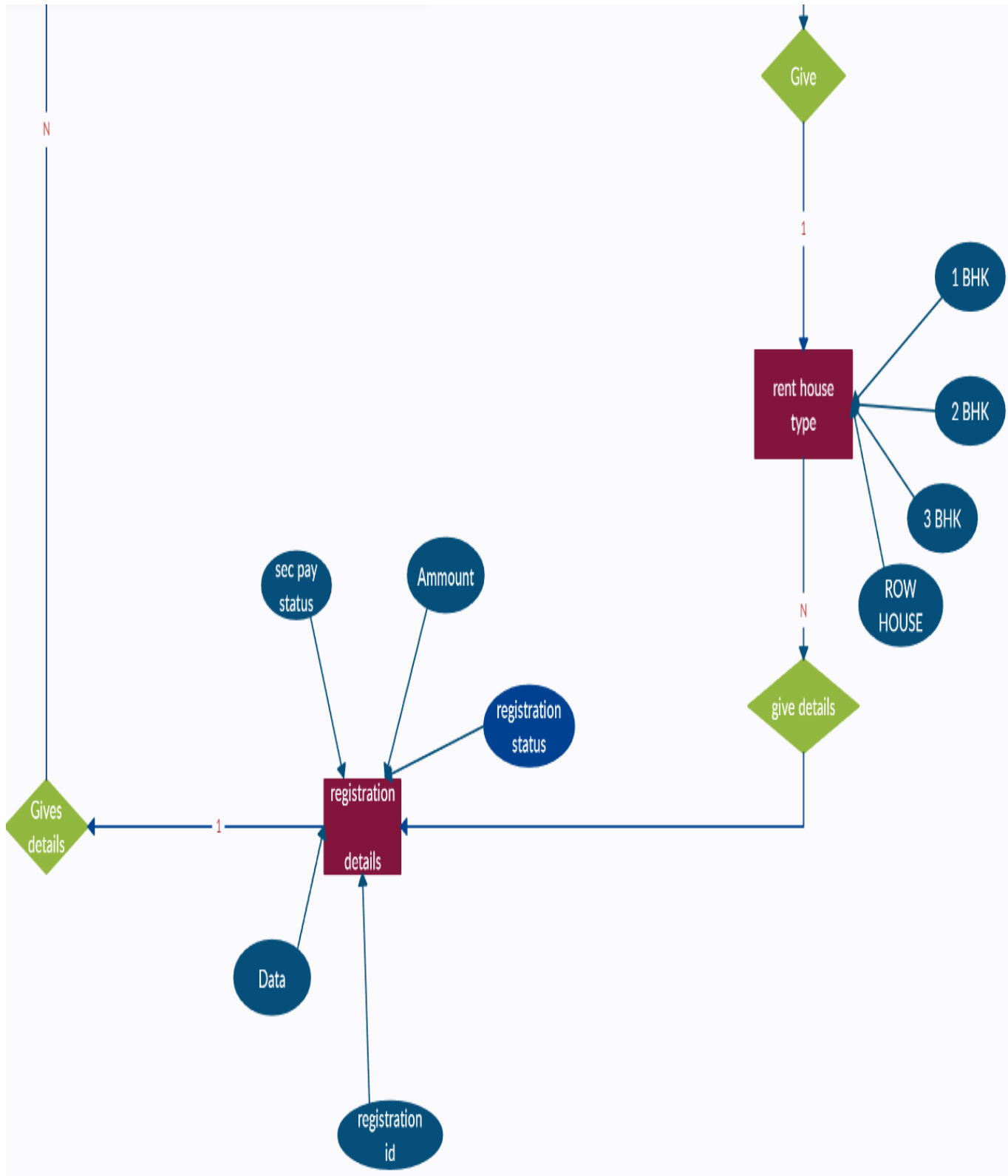
ERDs are essential in designing relational databases, providing a blueprint for database structure and organization. They help identify data requirements and ensure that the database will effectively support the needs of the application.

By clearly defining entities, attributes, and relationships, ERDs prevent redundancy, ensure data integrity, and facilitate efficient data retrieval. They are used by database designers, developers, and analysts during the planning and modeling stages of database development.

ERDs also aid in communication between stakeholders, providing a clear visual representation of the database structure. Tools like Microsoft Visio, Lucidchart, and draw.io are commonly used to create ERDs. Understanding ERDs is fundamental for anyone involved in database management, design, and implementation.



**Figure 4.2.1 tenant & owner ERD diagram**



**3.2.2 registration & renting ERD diagram**



## 4.3 Interface Design

We seek to minimize time and efforts needed by the user to do some action. So, the interface of the application should be clear, easy to use, easy to understand, and smooth. The following subsections show the designed views of the website and description of each view.

### Profile

After login successfully, the application will open the user's profile. The profile page contains user information.

#### • Signup page(register):

- 1- First input field: used for entering name.
- 2- Second input field: used for entering phone number.
- 3- Third input field: used for entering email.
- 4- Fourth input field is to select the Role of the user (Teacher OR place owner)
- 5- Fifth input field: used for entering the city of the user.
- 6- sixth input field: used for entering password.
- 7- Seventh input field: used for entering password conformation.
- 8 Signup button: Confirm registration process.
- 9- Signup with REFT link: Navigate to login page (if you already have an account).

#### • Log in Page:

- 1- First input field: used for entering the user email.
- 2- Second input field: used for entering password.
- 3- Login button: Confirm access to your account.
- 4- Signup with REFT link: Navigate to sign up page (if you don't have an account)



**Login in**

Email

password

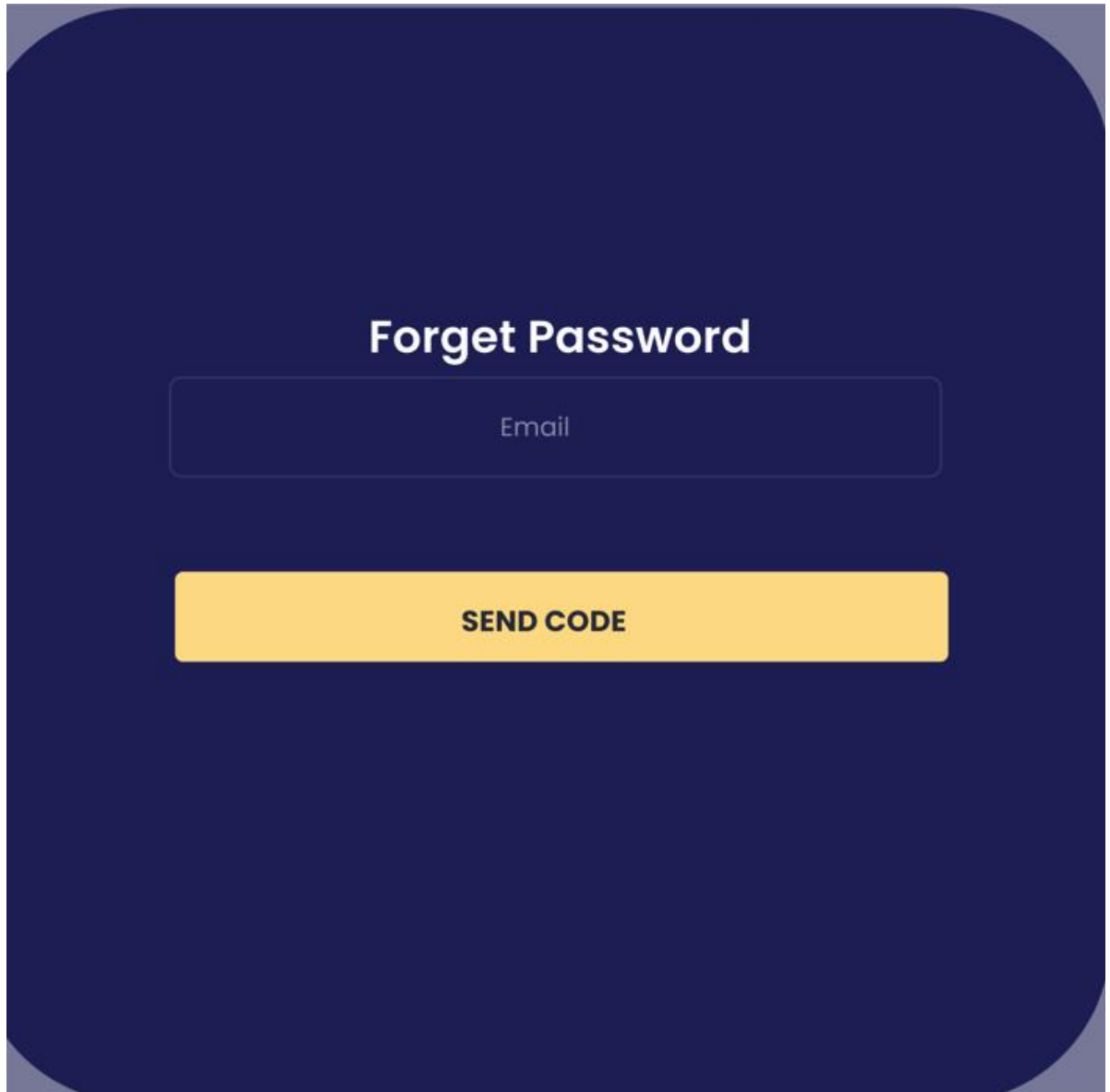
**login**

have an account? [Sign up](#)

**Figure 4.4 login page**

## Forget password

If the user forgets the password of his profile, he can click on forget password and enter the Email associated with his profile, and a code will be sent to his Email to reset the Password.

A dark blue rounded rectangle containing the text "Forget Password" in white. Below the title is a light blue rounded rectangle with the placeholder text "Email" in a lighter blue. Below that is a yellow rounded rectangle with the text "SEND CODE" in black.

**Forget Password**

Email

**SEND CODE**

Figure 4.5 forget password page

**Enter New Password**

Password

New password

**SAVE CHANGES**

Figure 4.6 enter a new password

## Home page

the home page contains: An introductory paragraph of the site & Our website goals.

2- The highest rated places & Reviews from our customers and FAQ.

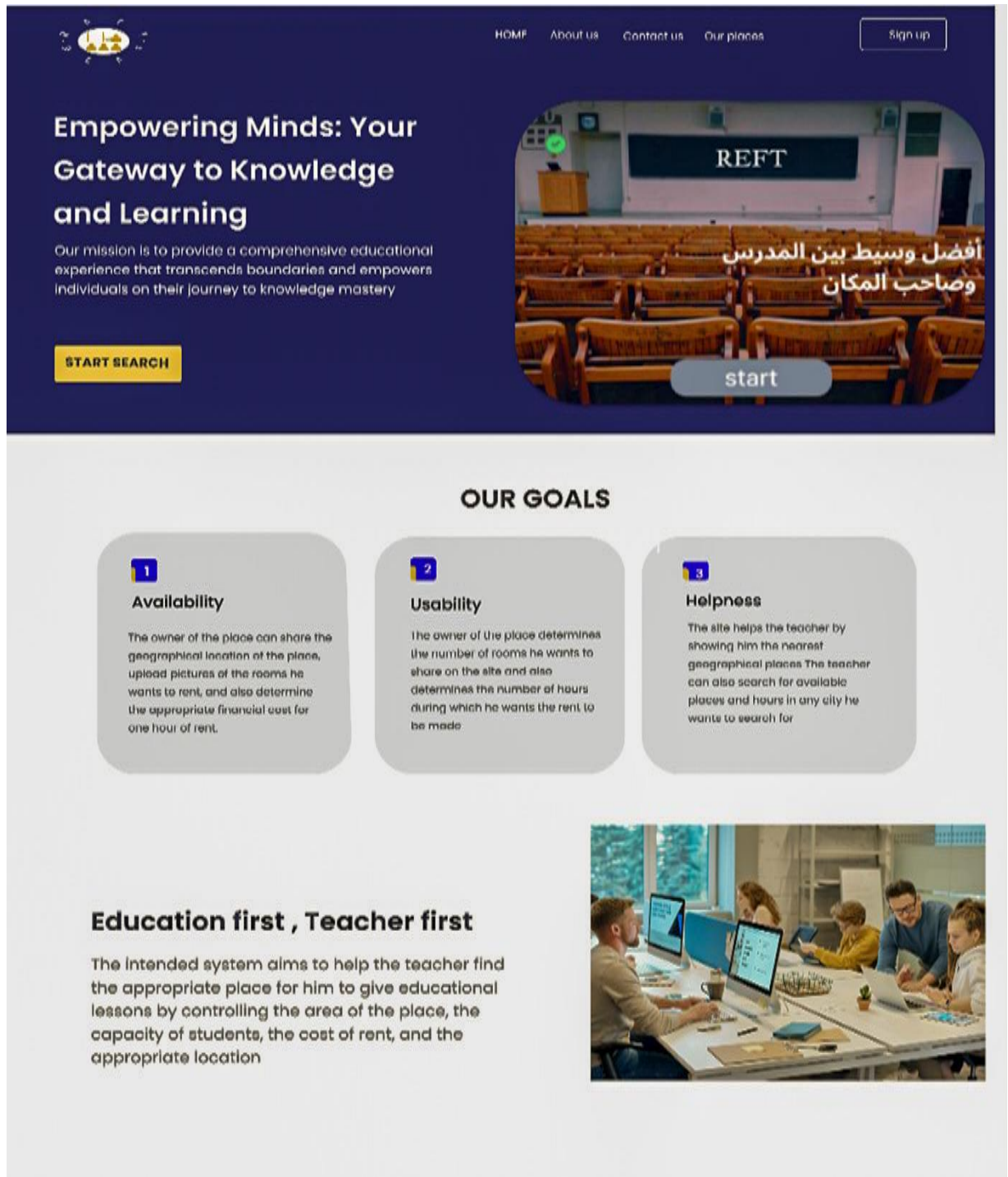


Figure 4.7.1 home page

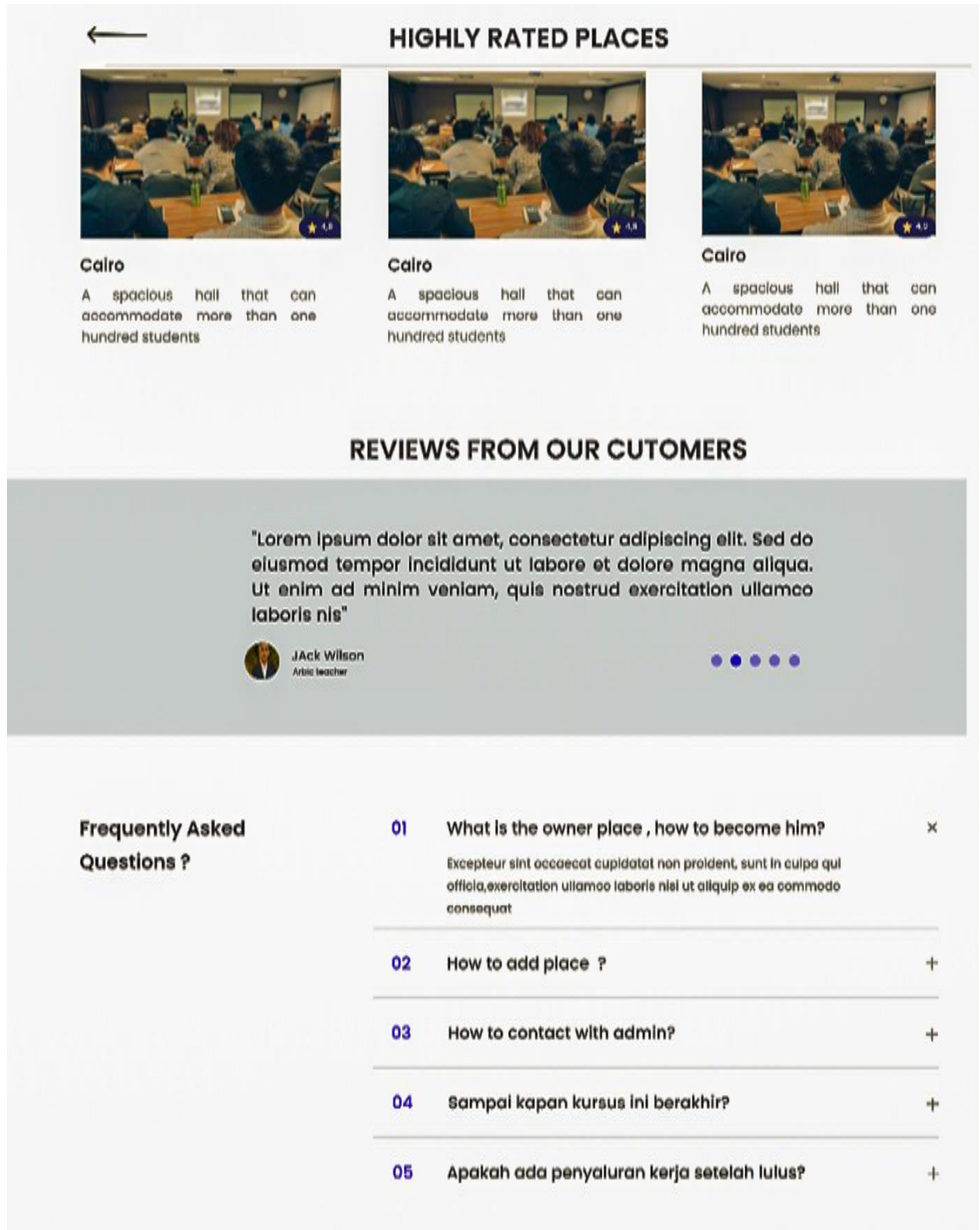


Figure 4.7.2 rest of home page

## Add place page


The owner of a place can add his property through this page in which the details of the place is filled in by the owner Upload video of the place.

---

### Add Hall

Place Name


Place Capacity

Place City 


Hour Price

Place Details

Add images of place

  
ADD IMAGE

Select property of place eg : place ownership contract



Select Video


submit 

Figure 4.8 Add hall page



## Place Details Page

This page contains the information of the place owner and the place details which are filled in by the owner of the place.

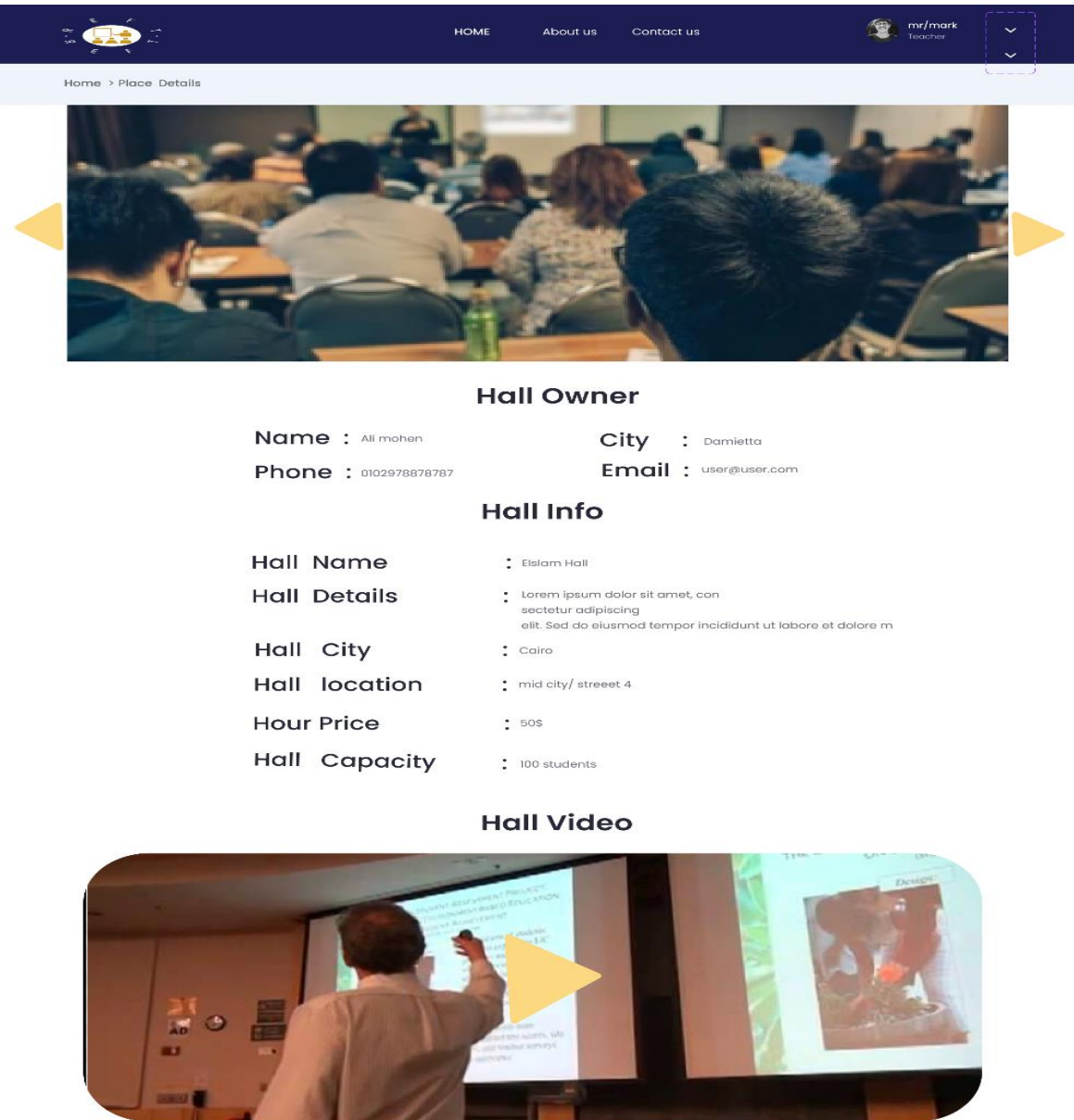
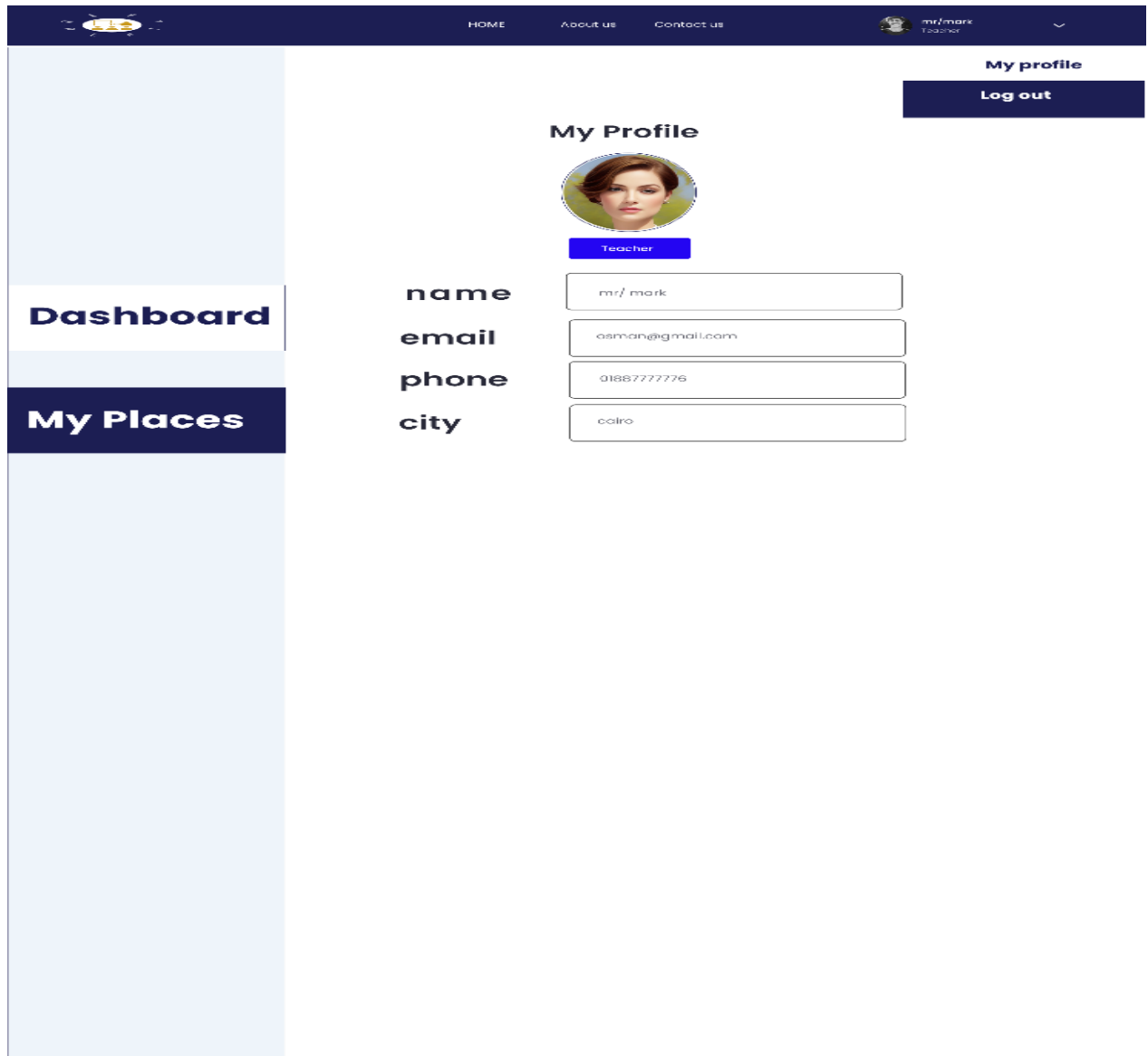


Figure 4.10 Hall details page

## User profiler:

In this page the information of the user is showed to him (the name of the user, his email, his city and his phone number) and he can also change his information.



The screenshot shows a web application interface for a user profile. On the left is a vertical sidebar with a light blue background, containing a 'Dashboard' link and a 'My Places' link. The main content area has a white background. At the top right, there is a navigation bar with links for 'HOME', 'About us', and 'Contact us', along with a user profile icon and the text 'mr/mark Teacher'. Below this, a 'My profile' button is visible. The 'My Profile' section features a circular profile picture of a woman, a blue 'Teacher' role button, and four input fields for 'name' (mr/ mark), 'email' (asman@gmail.com), 'phone' (0188777776), and 'city' (colro).


My Profile	
	
	<b>Teacher</b>
<b>name</b>	<input type="text" value="mr/ mark"/>
<b>email</b>	<input type="text" value="asman@gmail.com"/>
<b>phone</b>	<input type="text" value="0188777776"/>
<b>city</b>	<input type="text" value="colro"/>

Figure 4.11 Dashboard

## The owner places page:

On this page there are all the places that the owner added and were accepted and uploaded to the site

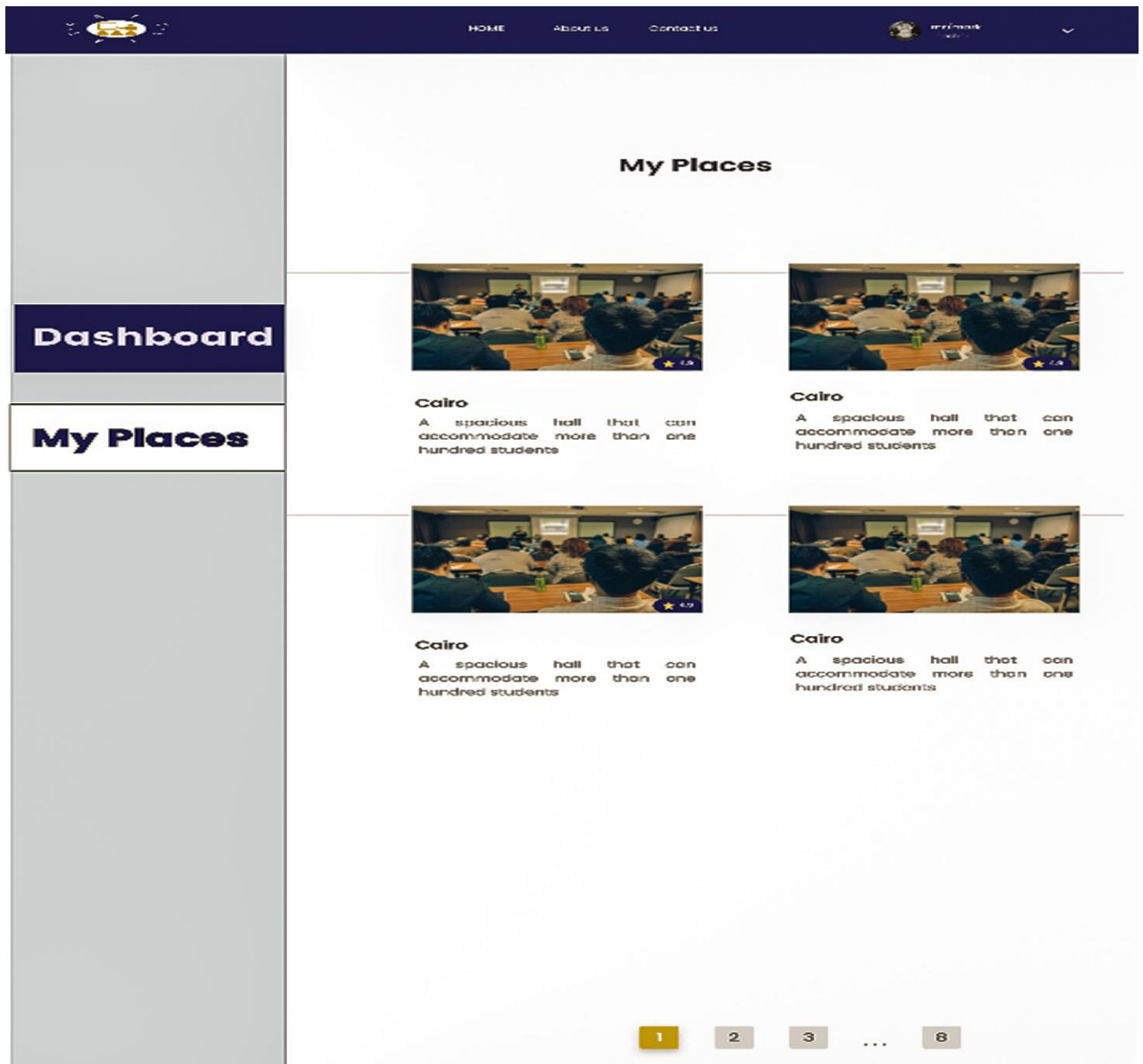


Figure 4.12 My places page

## About us page

In about us there is the **Company overview** A brief summary of what the organization does, its industry, and its main products or services.

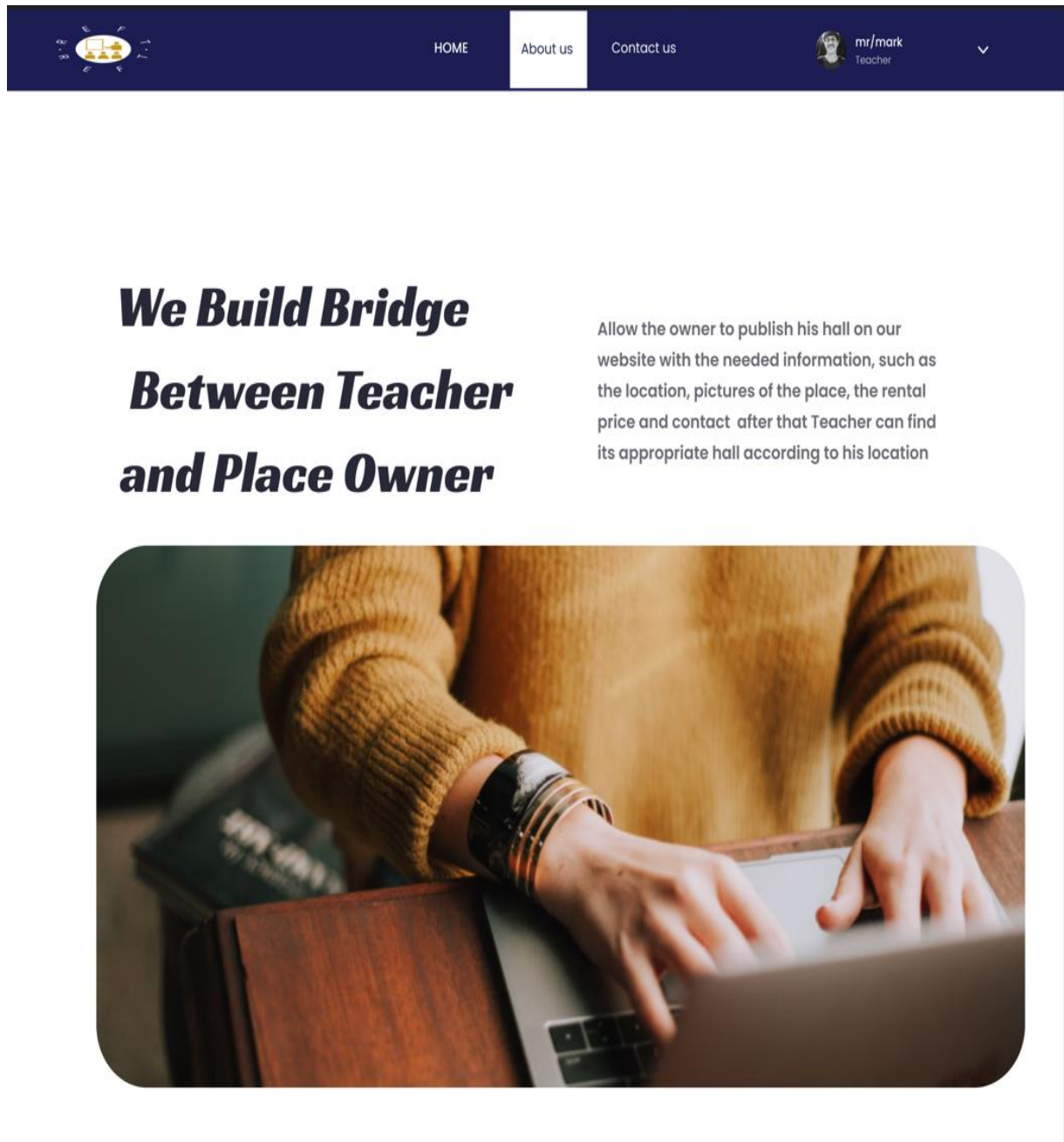




Figure 4.13 about us page

## contact us page


### GET IN TOUCH

 Phone

987654345667

 Email

osman@gmail.com

 Address

Damietta/ Elsaidy street 22

SEND MESSAGE

Figure 4.14 contact us page

# **Chapter 5**

## **System**

### **Implementation**

# Chapter 5: System Implementation

## 5.1 Technologies

### Front-end Development

Frontend development focuses on the user interface and user experience aspects of the system. It involves creating the visual elements that users interact with directly.

### Tools and Technologies

- HTML, CSS, and JavaScript: These core technologies are used to create the structure, style, and interactivity of web pages.
- React Bootstrap: The popular front-end framework, rebuilt with React - react-bootstrap.
- React: The library for web and native user interfaces
- Figma: A design tool for creating and prototyping user interfaces collaboratively.

## Objectives

- Develop an intuitive and visually appealing user interface.
- Ensure responsiveness across various devices and screen sizes.
- Implement interactive elements to enhance user engagement.

## Work flow

- Designing the User Interface: Using Figma, design the UI layout and interactive elements.
- Developing the Structure: Use HTML to create the basic structure of the web pages.
- Styling the Interface: Apply CSS and Bootstrap to style the web pages, ensuring a cohesive and responsive design.
- Adding Interactivity: Implement JavaScript and React Bootstrap to add dynamic and interactive elements to the web pages.
- Testing and Refinement: Continuously test the frontend to ensure usability and visual appeal, making refinements as needed.

## Back-end Development

Backend development involves server-side logic, database interactions, and application functionality. It focuses on the underlying systems that power the frontend interface.



## **Tools and Technologies**

- Express: Fast, unopinionated, minimalist web framework for Node.js
- PostgreSQL: A powerful and scalable database system for managing data.

## **Objectives**

- Develop robust server-side logic to handle user requests and data processing.
- Ensure data integrity, security, and performance.
- Integrate APIs to extend the functionality of the application.

## **Work Flow**

- Setting Up the Environment: Configure Express and PostgreSQL to create a development environment.
- Database Design: Design the database schema using PostgreSQL, ensuring data normalization and integrity.
- Implementing Models and Views: Develop Express models to represent database entities and views to handle user requests.
- Creating Templates: Use Express's template engine to create dynamic HTML templates.
- Integrating APIs: Connect various APIs to enhance the application's functionality.
- Testing and Debugging: Thoroughly test backend components to ensure they work as expected and debug any issues that arise.

## **Integration**

Integrating the frontend and backend involves connecting the user interface with server-side logic to create a seamless user experience.

## **Workflow**

- Frontend Design: Use Figma to design the user interface and prototype interactions.
- Frontend Implementation: Develop the UI using HTML, CSS, JavaScript, React Bootstrap, and React.
- Backend Development: Build the server-side logic using Express and manage data with PostgreSQL.
- API Integration: Connect various APIs to enhance functionality, such as maps, fonts, and icons.
- Testing and Deployment: Test the integrated system thoroughly and deploy it to a production environment.

## **5.2 Sample Application Codes**

After we have designed the system, we will map our design into implementation. Section 5.2 discusses the implementation process considering our objectives. Next in Section 5.3, we view samples of the code. Section 5.4 and section 5.5 discuss the testing process and result.

Login to application:

```
43     password: password,
44   }},
45 );
46 setLoading(false);
47 };
48
49 useEffect(() => {
50   if (loading === false) {
51     if (res.data) {
52
53       if (res.data.validationError) {
54         setLoading(true);
55         alert(res.data.validationError);
56         return;
57       }
58       if (res.data.token) {
59
60         localStorage.setItem('token', res.data.token);
61
62         localStorage.setItem('user', JSON.stringify(res.data.data));
63
64
65         setLoading(true);
66         setTimeout(() => {
67           window.location.href = '/';
```

Figure 5.1 Code for login page

Signup to application:

```
72     value={email}
73     onChange={onChangeEmail}
74   ></input>
75
76   <div className="input-wrapper">
77     <input
78       className="inputfield"
79       type={showPassword ? 'text' : 'password'}
80       placeholder="Password"
81       value={password}
82       onChange={onChangePassword}
83     />
84     <span
85       className="password-toggle-icon"
86       onClick={togglePasswordVisibility}
87     >
88       {showPassword ? (
89         <FontAwesomeIcon icon={faEye} />
90       ) : (
91         <FontAwesomeIcon icon={faEyeSlash} />
92       )}
93     </span>
94   </div>
95
96   <AcessButton txt="login" onClick={onSubmit} />
97   <AcessSwitch
```

Figure 5.2 Code for signup page

The code below for login check credentials for access website:

```
49   useEffect(() => {
50     if (loading === false) {
51       if (res.data) {
52
53         if (res.data.validationError) {
54           setLoading(true);
55           alert(res.data.validationError);
56           return;
57         }
58         if (res.data.token) {
59
60           localStorage.setItem('token', res.data.token);
61
62           localStorage.setItem('user', JSON.stringify(res.data.data));
63
64           setLoading(true);
65           setTimeout(() => {
66             window.location.href = '/';
67           }, 1000);
68
69         } else {
70
71         }
```

Figure 5.3 Code for credentials check

The code below to add place

```
frontend > src > pages > hall > JS HallAdd.js > HallAdd
15  function HallAdd() {
167      <input
168          className="inputfield-hall"
169          value={pdfName}
170          readOnly
171          placeholder="Select property of place eg : place ownership contract "
172          style={{ width: '100%' }}
173      />
174  </Col>
175  <Col xs="1" sm="1" md="1" lg="1" className="text-end">
176      <PdfChooser handleSelect={onChangePdf} />
177  </Col>
178  </Row>
179
180  <Row className="d-flex justify-content-center mt-2">
181      <Col xs="7" sm="7" md="7" lg="7" className="text-center">
182          <input
183              className="inputfield-hall"
184              placeholder="Select Video"
185              style={{ width: '100%' }}
186              value={videoName}
187              readOnly
188          />
189      </Col>
190      <Col xs="1" sm="1" md="1" lg="1" className="text-end">
191          <VideoChooser handleSelect={onChangeVideo} />
192      </Col>
```

Figure 5.4 Code for adding hall

The code samples below for admin check the validity:

```

22 <Col className="text-end">{status}</Col>
23
24 <Col className="text-end">
25   <FontAwesomeIcon icon={faCaretUp} flip="vertical" />{' '}
26 </Col>
27 </Row>
28 </p>
29
30 <div
31   className="auth-dropdown-content"
32   style={{
33     display: isActive ? 'block' : 'none',
34     fontWeight: 'bold',
35     fontSize: '20px',
36     color: '#282938',
37   }}
38 >
39   <div className="item" onClick={(e) => changeStatus('true')}>
40     Accept
41   </div>
42   <div className="item" onClick={(e) => changeStatus('false')}>
43     Refuse
44   </div>
45 </div>

```

Figure 5.5 Code for checking validity

The code below is about the searching process:

```

17 function Halls() {
47
48   const HandleChangeCity = (v) => {
49     const city = v.target.id;
50     const isChecked = v.target.checked;
51
52
53
54     if (city === 'all') {
55       setSelectedCity(isChecked ? [city] : []);
56       onChangeCity({ city: isChecked ? [city] : [] });
57     } else if (!isChecked) {
58       window.location.reload();
59     } else {
60       const updatedSelectedCity = isChecked
61         ? [...selectedCity, city]
62         : selectedCity.filter((selected) => selected !== city);
63       setSelectedCity(updatedSelectedCity);
64       onChangeCity({ city: updatedSelectedCity });
65     }
66   }
67

```

Figure 5.6 Code for searching process

## 5.3 System Testing

System Testing means testing the system as a whole. All the modules/components are integrated in order to verify if the system works as expected or not this plays an important role in delivering a high-quality product. We provide here white box and black box testing.

	#	Test case	Expected result	Actual Result
	1	Login with valid user name and password	User logs in successfully and the user profile page will be available. The user is marked as an attendant in the database. User can browse home page and choose between different services in app.	success
	2	Click on forget password.	Sending request to user's email, writing new password and save it in database and delete the first one.	success
	3	The owner add place	. After the owner has uploaded the place. the place should be moved to the admin list	success
	4	The place appears in the admin list	Admin see the places uploaded by the owners	success
	5	The place appears in REFT site after admin approval	When the admin accepts the place. it should be appearing for the teachers	success
	6	The place appears in search process	In search process, the place should appear if the search key words are related	success

7	More places should appear in the owner dashboard profile	All places added by one owner should be appears in his dashboard profile	success
8	User information is showed in the user profile	The user information that he had filled in it in sign up process, appears in the user dashboard	success
9	User can change his information	the user changes in his profile are updated in REFT site	success

Table 5.1: black box testing

#	Test case	Expected result	Actual result
1	Enter an invalid email or password.	Display message wrong email or password	success
2	Login with either email or password.	Display message require this field.	success
3	Check functionality of forget password.	The new password replaced with the old password in database.	success
4	If we enter some wrong letters before login.	There is <b>cancel</b> button available to erase the text.	success
5	If we press log out.	We couldn't see our profiles	success
6	If the place details are empty	Wrong message appears as he should fill in all the details	success

Table 5.2: white box testing



## 5.4 Results

Services provided by our REIT system. It helps teachers share the same place, which leads to optimal utilization of the place and also shares the financial cost between them based on the number of hours for each of them.

Any lecturer can rent an hour or two according to his need for a small financial price. In contrast, in the current situation, he can only do that in huge halls and at a high price.

The teacher can easily search for a place to teach in any city within Egypt

The owner can upload the place to our website, and thus this site will be displayed to a large number of teachers, larger than the normal case that relies on brokers.

The site has conditions regarding the places advertised on the site, thus ensuring that these places are of quality and suitable for the purpose of education.

## 5.5 Achieved Goals

We managed to build our system using simple components. The system achieved all its basic goals.

### **The First Goal: the place is shared by more than one teacher**

More than one teacher can rent the place according to his needs successfully

### **The second Goal: owner can add places.**

The owner can add a place and filling in details of the place and the uploaded it.

### **The third Goal: the place will be uploaded after the admin review and approval**

After the owner uploads the place, the place request will be sent to the admin and after the admin review and approval the place will be uploaded.

#### **The fourth Goal: reservation the place as you like**

The teacher can reserve as he likes from one hour to months according to his needs.

#	Goal	Achieved	Notes
1	the place is shared by more than one teacher	Yes	—
2	owner can add places	Yes	—
3	the place will be uploaded after the admin review and approval	Yes	—
4	reservation the place as the teacher needs	Yes	—

Table 5.3: achieved goals.

# **Chapter 6**

# **Conclusion**

# **And**

# **Future Work**

# **Chapter 6: Conclusion and Future Work**

## **6.1 Results**

The Reft website aims to allow teachers to rent suitable places for them to teach to students by renting the number of hours the teacher needs per day throughout the semester or the entire year, which contributes to reducing the teacher's expenses and thus reducing the prices of private lessons in our country, Egypt.

At the same time, it also aims to increase the profits of the property owner or hall owner by providing a larger number of teachers in the same place but distributed over the hours of the day.

## **6.2 Future work**

There is a lot of work to be done to make Reft a great educational platform and the first thought for teachers and owners of educational properties.

These are some of the things that we will work on developing in the future.

1- Create an artificial intelligence bot that recommends places to the teacher based on questions he asks the teacher

2- Create a community for teachers to communicate with each other

# References

- [1] React Bootstrap  
<https://react-bootstrap.netlify.app/>
  
- [2] React  
<https://react.dev/blog/2023/03/16/introducing-react-dev>
  
- [3] Node.js  
<https://nodejs.org/>
  
- [4] Express.js  
<https://expressjs.com/>
  
- [5] PostgreSQL  
<https://www.postgresql.org/>
  
- [6] Dipta Voumick, Prince Deb, Sourav Sutradhar, Mohammad Monirujjaman Khan. An Online Based Smart House Renting Web Application.  
<https://www.scirp.org/journal/paperinformation?paperid=110723>
  
- [7] Mandeep Katre (Professor), Kumar Abhay Gupta, Shipra Katiyar, Saumya Shahi, Shreya Awasthi. A Review Paper on PG Recommendation System.  
[https://www.researchgate.net/publication/370944421\\_A\\_Review\\_Paper\\_on\\_PG\\_Recommendation\\_System](https://www.researchgate.net/publication/370944421_A_Review_Paper_on_PG_Recommendation_System)
  
- [8] Murahari Prithvi Yash, Chinmay Choudhary, Akanksha Lakra, Swati Dewangan. RentoAxis: Android App for Paying Guest Management.  
<https://www.jetir.org/papers/JETIR1812A23.pdf>

- [9] F. H. Hassan, "Heuristic search methods and cellular automata modeling for layout design," Ph.D dissertation, Sch. of Info. Sys, Comp. and Math., Brunel Univ., UK, 2013.
- [10] S. Sarmady, F. Haron, and A. Z. H. Talib, "Modelling groups of pedestrians in least effort crowd movements using cellular automata," in Proc. 2009 2nd Asia International Conference on Modelling & Simulation, Bali, Indonesia, 2009, pp. 520-525.
- [11] V. J. Blue, and J. L. Adler, "Cellular automata micro-simulation of bi-directional pedestrian flows," J. Transportation Research, pp. 135-141, 2000.
- [12] Malan, R., & Bredemeyer, D. (2001). Functional requirements and use cases. *Bredemeyer Consulting*.  
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.436.4773&rep=rep1&type=pdf>
- [13] Dinh-Trong, T. T., Ghosh, S., & France, R. B. (2006, November). A systematic approach to generate inputs to test UML design models. In *2006 17th International Symposium on Software Reliability Engineering* (pp. 95-104). IEEE.  
<https://ieeexplore.ieee.org/abstract/document/4021975>

# Appendix A

## DVD Contents

The DVD of the project contains the following:

1. Full code of the web application
2. Documentation of the project in PDF format
3. Presentation of the project

Please note that you'll need to learn React and JavaScript programming languages to understand the code. You can view the official documentation of each language on its website on the internet or you can watch videos if you like (i.e. YouTube tutorial). You will also need to install Visual Studio Code on your machines to run the code.