



**Al-Imam Muhammad Bin Saud Islamic University**  
**College of Computer and Information Sciences**

Department of Computer Science

CS346 (Course project)

June 6th, 2023

# Office hour



Name	Email	ID
Wea'am Alghaith	wksalghait@sm.imamu. edu.sa	4400233 06
Khloud Alnufaie	khaalnufaie@sm.imamu. .edu.sa	4400206 17
Raghad Albosais	rkaalbosais@sm.imamu. edu.sa	4400202 09
Alhanouf Almansour	aalmansour83@sm.ima mu.edu.sa	4400191 83



## Contents

1. Introduction.....	4
1.1 Problem Definition.....	4
1.2 Aims.....	4
1.3 Objectives .....	4
2. Front End .....	5
.....	5
2.1 User Interface (UI).....	6
<b>2.1.1 Signup Function .....</b>	<b>6</b>
<b>2.1.2 Sign-in Function .....</b>	<b>7</b>
<b>2.1.3 Booking Function .....</b>	<b>7</b>
<b>2.1.4 Display Teacher's Information .....</b>	<b>8</b>
<b>2.1. 5 Cancel Reservation .....</b>	<b>10</b>
.....	11
<b>2.1.6 Modify Information Function .....</b>	<b>11</b>
.....	11
.....	11
.....	11
2.2 Implementation.....	12
2.2.1 General structure for all pages .....	13
<b>2.2.2 Sign up Page .....</b>	<b>14</b>
<b>22.2 Sign in Page. ....</b>	<b>14</b>
<b>2.2.3 Main Page .....</b>	<b>15</b>
<b>2.2.4 Booking Page .....</b>	<b>16</b>
<b>2.2.5 My Booking Page .....</b>	<b>17</b>
<b>2.2.6 My Information Page.....</b>	<b>18</b>
<b>2.2.7 About Page.....</b>	<b>18</b>
3. Back End.....	19
.....	19
3.1 Implementation .....	20
<b>3.1.1 Models Folder and Main File.....</b>	<b>20</b>
<b>3.1.2 Server File.....</b>	<b>24</b>



## List of figures:

Figure 1. Sign up function .....	6
Figure 2. Sign in function .....	7
Figure 3. Booking function .....	8
Figure 4. Display teacher information .....	9
Figure 5. Cancel reservation function .....	10
Figure 6. Modify my information .....	11
Figure 7. Front end files structure .....	12
Figure 8. General HTML structure for all pages .....	13
Figure 9. HTML for sign up .....	14
Figure 10. HTML for sign in .....	14
Figure 11. HTML for main page.....	15
Figure 12. HTML for booking page.....	16
Figure 13. HTML for my booking page.....	17
Figure 14. HTML for my information page.....	18
Figure 15. HTML for about page.....	18
Figure 16. Back end file structure.....	19
Figure 17. Data schema for teacher. ....	20
Figure 18. Data schema for student. ....	21
Figure 19. Data schema for reservation. ....	22
Figure 20. Teacher, Student and Reservation classes. ....	23
Figure 21. initialization for Server.js .....	24
Figure 22. the route for sin in page. ....	25
Figure 23. The route to the main page .....	25
Figure 24. The route to my booking page.....	26
Figure 25. Booking post request. ....	27
Figure 26. Cancel reservation route. ....	28
Figure 27. Sign in post request. ....	29
Figure 28. Sign up post request.....	30



# 1. Introduction

An office hour application offers students to schedule an appointment with their teachers to discuss unclear points for them, discuss the exam results, or ask for guidance. In addition, it provides students with important information about their teachers, such as office numbers, email addresses, and schedules.

## 1.1 Problem Definition

Students sometimes have difficulty reaching a faculty member's information, such as their office number, schedule, and email, which may not be updated. Also, there is a problem when the teacher is available during office hours, but sometimes no student attends. Additionally, if many students are around a teacher's office, it may be uncomfortable for both parties.

## 1.2 Aims

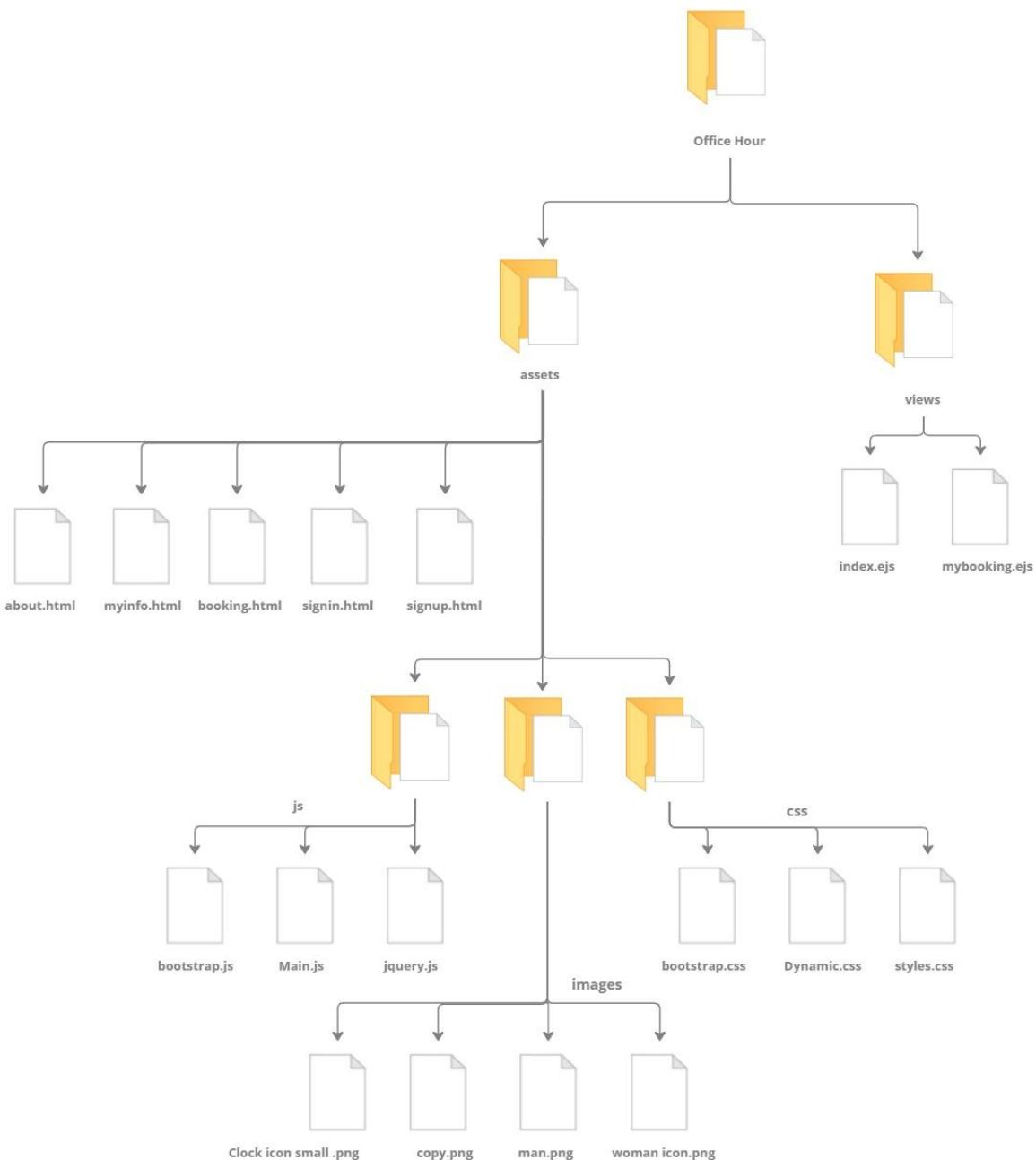
- The students find all the important information about their teacher in one place.
- Informing the teacher about the reservation of the student during office hours, which makes him present in his office.
- Organizing and facilitating communication between the teacher and the student.

## 1.3 Objectives

- Display the teacher's name, office number, official email, and schedule in one page.
- Inform the teacher of the student's attendance by sending a notification after the student has made an appointment.
- Through the application, students will be able to schedule appointments with their teachers in order to facilitate communication.
- This application organizes communication between the teacher and the student by limiting the number of students that can reserve in a one-hour period.



## 2. Front End





## 2.1 User Interface (UI)

In this section we present the UI for each function and the relationship between pages.

### 2.1.1 Signup Function

البريد الجامعي:  
mosalomar@sm.imamu.edu.sa

كلمة المرور:

هل نسيت كلمة المرور؟

دخول

البريد الجامعي:  
mosalomar@sm.imamu.edu.sa

كلمة المرور:

تأكيد كلمة المرور:

اسم الأول: مها

اسم الأب: شير

اسم العائلة: سليمان

العمر:

تسجيل

خطىء

**Step 1:** The user will Click on "تسجيل جديد" (Sign Up) button.

**Step 2:** The user will enter (First name, Second name, Third name, Fourth name,

Official email, password, confirm password).

**Step 3:** The user clicks "تسجيل" (Sign Up) button.

Figure 1. Sign up function.

## 2.1.2 Sign-in Function

ساعة مكتبة



تسجيل دخول

البريد الجامعي:

wksalghait@sm.mamu.edu.sa

كلمة المرور:

.....

هل سوت كلمة المرور؟

إغلاق

**Step 1:** The user enters email and password.  
**Step 2:** The user clicks on "دخول".

*Figure 2. Sign in function*

### 2.1.3 Booking Function

The screenshot displays the 'Maktabah' mobile application's main interface. At the top center is a digital clock showing 12:00. To the left of the clock is the text 'ساعة مكتبة' (Library Clock). On the right side of the clock are three small circular icons representing different features.

Below the clock, there is a horizontal navigation bar with the following items from left to right: 'الصفحة الرئيسية' (Main Page), 'عن مكتبة مكتبة' (About Maktabah Library), 'جورنالي' (Journalist), 'تعديل بياناتي' (Edit Profile), and 'توكيل بيتنا' (Our House Agent).

The main content area features a grid of six silhouette icons arranged in two rows of three. The icons represent different roles: 'بنت شاعر' (Poet Girl) with a female silhouette, 'مداد صحف' (Press Pen) with a male silhouette, 'زب الفتن' (Faction Reader) with a male silhouette, 'حالة نمر' (Leopard State) with a female silhouette, 'دلة تحرير' (Editorial Docket) with a male silhouette, and 'مشن الخطيب' (Speaker Reporter) with a male silhouette. A cursor arrow is positioned over the 'مداد صحف' icon.

Below this grid is another section featuring a digital clock at the top center, with the text 'ساعة مكتبة' (Library Clock) to its left. This section includes a horizontal navigation bar with 'الصفحة الرئيسية', 'عن مكتبة مكتبة', 'جورنالي', 'تعديل بياناتي', and 'الوقت' (Time) on the right.

The bottom portion of the screen contains several filter options arranged in a grid:

- 'الاليقون' (Icon) with a dropdown menu containing 'CS-242'.
- 'الوقت' (Time) with a dropdown menu containing '371'.
- 'الاليقون' (Icon) with a dropdown menu containing 'الاولين' (Firsts).
- 'الوقت' (Time) with a dropdown menu containing '12:00-12:30'.

At the very bottom of the screen are two large, rounded rectangular buttons: 'رجوع' (Back) on the left and 'جزء' (Part) on the right, with a cursor arrow pointing towards the 'جزء' button.



imam.cs.oh@gmail.com  
To: Weaam Khaled Saleh Alghaith

Sun 04/06/2023 05:13

ساعة مكتبة

استاذنا العزيز نفيذك علما انه  
تم حجز موعد في ساعتك المكتبة

بريدة المكتبة المقرر اليوم الجمعة الوقت 12:00-  
إيام عدد CS- 371 الاثنين 12:30 صلاح ديفit  
wksalghaith@sm.imamu.edu.sa

شكرين لك وشك

© 2023. ساعة مكتبة. All rights reserved.

Figure 3. Booking function.

**Step 1:** The user will click on any teacher element.

**Step 2:** The user will click on "حجز".

**Step 3:** The user will fill four fields by choosing one option from the drop-down menu provided by the system.

**Step 4:** After the user fills all fields (Day, Time, course, section number), then he/she will click "حجز" button.

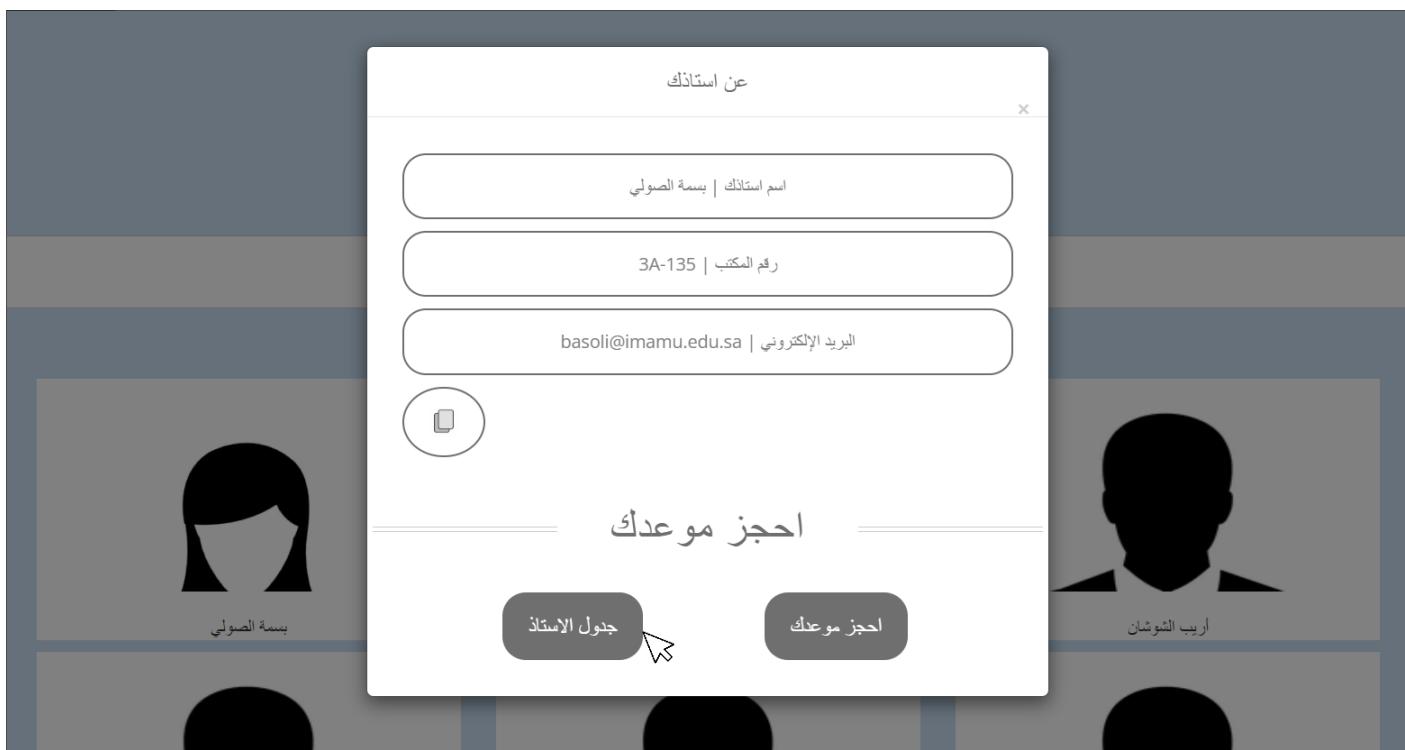
**Note:** the teacher will receive a message on his email notify him about the student reservation.

## 2.1.4 Display Teacher's Information

ساعة مكتبة

الصفحة الرئيسية عن ساعة مكتبة حجوزاتي تعديل بياناتي

 بسنة المصاوي	 هدى الله صبيح	 أربيب الشوشان
 هدى الله البراد	 وليد النويصر	 سلطان الخطاطني



The PDF document displays a weekly schedule for Basmah Alsouly, listing her office hours and availability across four days (Sunday through Wednesday). The schedule includes room numbers and specific times for each session.

	1 <sup>st</sup> 7:30 - 8:20	2 <sup>nd</sup> 8:25-9:10	3 <sup>rd</sup> 9:20-10:10	4 <sup>th</sup> 10:15-11:05	5 <sup>th</sup> 11:10-12:00	Break 12:00-12:30	6 <sup>th</sup> 12:30-13:20	7 <sup>th</sup> 13:25-14:15	8 <sup>th</sup> 14:20-15:10
Sunday									
Monday	CS242 374 1C-123	CS242 374 1C-123	CS242 374 1C-123	CS242 374 1C-123		OH 2C-324	CS346 372 2C-324	CS346 372 2C-324	
Tuesday	CS242 374 1C-123	CS242 374 1C-123	CS242 374 3C-114	CS242 374 1C-123	OH/ Student Advising		CS346 371 3C-224	CS346 371 3C-224	
Wednesday				CS346 372 2C-324	OH/ Student Advising	OH/ Student Advising	CS346 371 3C-224		
Thursday									

Below the schedule, there are buttons for navigating through the pages: '+', '🔍', '−', '1 / 1', and 'صفحة' (Page).

Figure 4. Display teacher information.



## 2.1. 5 Cancel Reservation

The screenshot shows the library management system's user interface. At the top, there is a navigation bar with Arabic text: "ساعة مكتبة" (Library Time) and "تعديل بياناتي" (Edit Profile). Below the navigation bar is a list of user profiles, each with a placeholder profile picture and a name:

- سمة المصوتي (Samaa Al-Mashti)
- هدأة صبيح (Hudaat Sabihi)
- أرب الشوشان (Arab Al-Shushan)
- هدأة البراك (Hudaat Al-Barak)
- وليد الورسر (Waleed Al-Warsar)
- سلطان الخطاطني (Sultan Al-Khatatni)

Below this is another section with a clock icon and Arabic text: "هل ترغب إلغاء الحجز؟" (Do you want to cancel the booking?). It includes fields for "الوقت" (Time), "اليوم" (Day), and "اسم الاستئجار" (Booking Name), with the value "سمة المصوتي" (Samaa Al-Mashti) entered.

**Step 1:** The user will click on "حجوزاتي" in the navigation bar.

**Step 2:** The user will click on "إلغاء" button.

**Note:** the teacher will receive a message on his email notify him about the student cancelation.

An email notification is shown, addressed to "imam.cs.oh@gmail.com" (To: Weaam Khaled Saleh Alghaith). The subject line is "استأننا العزيز نفيدك علما أنه تم إلغاء موعد في ساعتك المكتبة". The email body contains the reservation details:

الوقت	اليوم	اسم الاستئجار
12:00-12:30	الاثنين	سمة المصوتي

At the bottom, there is a message of thanks: "شكرين لك وفتوك".

At the very bottom of the email, there is a copyright notice: "© 2023 ساعه مكتبه . All rights reserved."

Figure 5. Cancel reservation function.



## 2.1.6 Modify Information Function

The screenshot shows a user interface for modifying account information. At the top, there is a navigation bar with Arabic text: 'ساعة مكتبية' (Library Clock), 'الصفحة الرئيسية' (Home Page), 'عن ساعه مكتبيه' (About Library Clock), 'محوزاتي' (My Assets), and 'تعديل بياني' (Edit My Info), which is highlighted with a cursor. Below the navigation bar is a grid of six profile pictures, each with a name below it: 'بسنة الصولي', 'هدى الله صبيح', 'أربيب الشوشان', 'هدى الله البراك', 'وليد الوبيصر', and 'سلطان الخطاطني'. In the middle section, there is a form for changing personal details. The fields and their corresponding values are:

الإسم الأول:	ونم
الإسم الثاني:	هلال
اسم الأب:	صالح
اسم العدة:	الغوث
البريد الجامعي:	wksalghait@sm.imamu.edu.sa
كلمة المرور:	*****
تأكيد كلمة المرور:	*****

A 'حفظ' (Save) button is located at the bottom of the form.

Figure 6. Modify my information.

**Step 1:** The user will click on "تعديل بياني" in the navigation bar.

**Step 2:** The user change any of these information (first name, second name, third name, fourth name, Official email, password, confirm password).

**Step 3:** The user clicks "حفظ" button.



## 2.2 Implementation

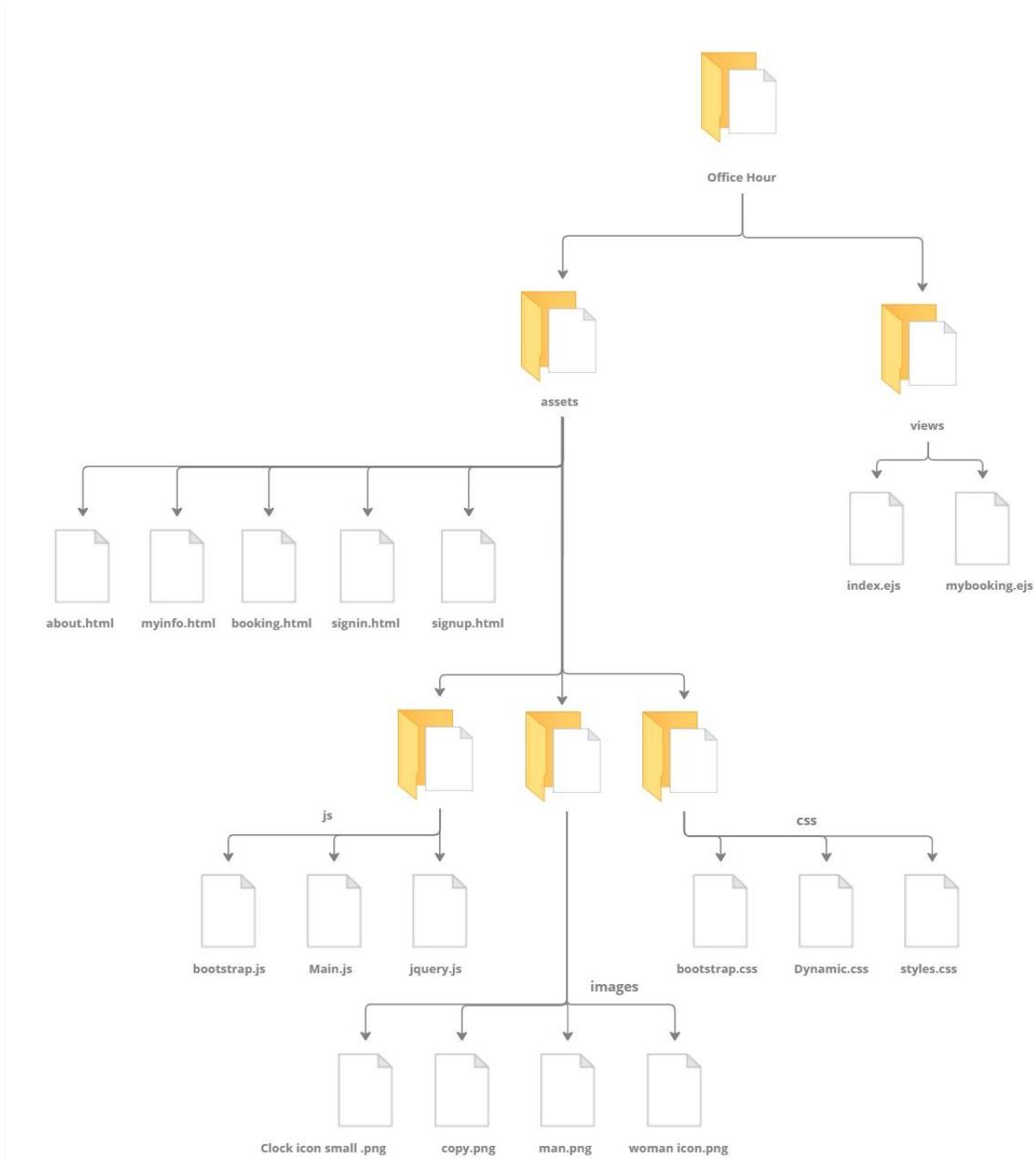


Figure 7. Front end files structure.



## 2.2.1 General structure for all pages

Page structure

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">

    <!-- The viewport varies with the device and will be smaller on a mobile phone than on a computer screen. This gives the browser instructions on controlling the page's dimensions and scaling. -->
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- Provide a short description of the page -->
    <meta name="description" content="">

    <!-- the document's title that is shown in a browser's title bar or a page's tab-->
    <title>(Page Name)</title>

    <!-- the icon that will appear in a page's tab -->
    <link rel="shortcut icon" href="images/Clock icon small .png">

    <!-- Bootstrap -->
    <link href="css/bootstrap.css" rel="stylesheet">

    <!-- Custom styles -->
    <link rel="stylesheet" href="css/styles.css">
</head>

<body class="home">
    <header id="header"> <!-- the register page header (start)-->

        <div id="head"> <!-- the name of our website and the logo (start)-->
            <h1 id="logo" class="text-center">
                <span class="title" style="color:#00F0F0;">ساعة مكتبة</span>
                
            </span>
        </h1>
        </div> <!-- the name of our website and the logo (end)-->

        <nav class="navbar navbar-default" > <!-- the navigator (start)-->
            <div class="container-fluid">
                <!-- the content of navigator -->
                <div class="navbar-collapse collapse">
                    <ul class="nav navbar-nav">
                        <li><a href="/" style="color:#00F0F0; font-size: medium;">الصفحة الرئيسية</a></li>
                        <li><a href="about.html" style="color:#00F0F0; font-size: medium;">عن ساعة مكتبة</a></li>
                        <li><a href="#" style="color:#00F0F0; font-size: medium;">موزعات</a></li>
                        <li><a href="mybooking" style="color:#00F0F0; font-size: medium;">حجز مواعيد</a></li>
                        <li><a href="myinfo.html" style="color:#00F0F0; font-size: medium;">بيانات ملائكة</a></li>
                    </ul>
                </div>
            </div>
        </nav> <!-- the navigator (end)-->
    </header> <!-- the register page header (end)-->

    <main id="main">
        {Page content}
    </main>

    <footer id="footer" class="topspace">
        <div class="container">
            <div class="row text-right">
                <h3 class="widget-title">اتصل بنا</h3>
                <p><a href="mailto:imam.cs.oh@gmail.com">imam.cs.oh@gmail.com</a></p>
            </div>
        </div> <!-- /row of widgets -->
    </footer>

    <footer id="underfooter">
        <div class="container">
            <div class="row">
                <div class="widget-body">
                    <p>Copyright © 2023, Office Hour<br></p>
                </div>
            </div> <!-- /row of widgets -->
        </div>
    </footer>

    <!-- JavaScript lines are placed at the end of the document so the pages load faster -->
    <script src="js/jquery.js"></script>
    <script src="js/bootstrap.js"></script>
    <!-- Main JavaScript file -->
</body>
</html>
```

localhost:8080/page%20structure.html

ساعة مكتبة

الصفحة الرئيسية     عن ساعة مكتبة     حجز مواعيد     تعديل بياناتي

Copyright © 2023, Office Hour

Note: Please note that all pages have the same header, body header, and body footer structure. The only difference between pages is the main body, so we will illustrate this part for each separately.

Figure 8. General HTML structure for all pages.



## 2.2.2 Sign up Page

```
signup.html

<main id="main">

<input class="positioning" id="register" type="button" value="رجوع"
    onclick="window.location.href='signin.html';" style="left: 88%; top: -180px;" >
<form style="width:850px; height:530px; left:1%; " method="post" class="container box
positioning" action="/sign_up">

<div>
    <div class="positioning" style="left:-10%;" >
        <h5 class="textIn">الاسم الأول:</h5>
        <input id= "name1" type="text" name="Fname" placeholder="الاسم الأول">
        <h5 class="textIn">الاسم الثاني:</h5>
        <input id= "name2" type="text" name="Sname" placeholder="الاسم الثاني">
        <h5 class="textIn">الاسم الثالث:</h5>
        <input id= "name3" type="text" name="Tname" placeholder="الاسم الثالث">
        <h5 class="textIn">اسم العائلة:</h5>
        <input id= "name4" type="text" name="Lname" placeholder="العائلة">
    </div>
<div class="positioning" style="right:100%; bottom: 480px">
    <h5 class="textIn">البريد الجامعي:</h5>
    <input id= "Email" type="email" name="email" placeholder="mosalomar@sm.imamu.edu.sa">
    <h5 class="textIn">كلمة المرور:</h5>
    <input id= "password#1" type="password" name="password1">
    <div id="submitInfo" class="positioning" style="display: block; left: -24%; bottom:
-50px;">
        <input id="submit" type="submit" value="تسجيل" name="submit1" style="width: 130px;">
    </div>
</div>
</div>
</form>
</main>
```

الاسم الأول: mosalomar@sm.imamu.edu.sa  
الاسم الثاني:  
كلمة المرور:  
البريد الجامعي: mosalomar@sm.imamu.edu.sa  
اسم العائلة:  
الرمز:

Figure 9. HTML for sign up.

## 2.2.2 Sign in Page.

```
signup.html

<main id="main">
<input class="positioning" id="register" type="button" value="تسجيل جديد"
    onclick="window.location.href='signup.html';" style="left: 88%; top: -180px;" >
<form method="post" class="container" action='/signin'>
    <div class="text-right box">
        <h5 class="textIn">البريد الجامعي:</h5>
        <input id= "username" type="email" name="username"
            placeholder="mosalomar@sm.imamu.edu.sa">
        <h5 class="textIn">كلمة المرور:</h5>
        <input id= "password" type="password" name="password">
        <div id="submitInfo" class="positioning" style="display: block; border: 1px solid black; width: 130px; margin-left: auto; margin-right: 0;">
            <input id="submit" type="submit" value="دخول" name="submit" >
        </div>
    </div>
</form>
</main>
```

البريد الجامعي: mosalomar@sm.imamu.edu.sa  
كلمة المرور:  
دخول:

Figure 10. HTML for sign in.



## 2.2.3 Main Page

```
index.ejs
```

```
<main id="main">
  <div class="container">
    <div class="row section recentworks topspace">
      <%teachers_list.forEach(teacher => {%
        <%console.log(teacher)%>
      }%>
    </div>

    <div class='col-lg-4'>
      <a class='thumbnail' href='/index' data-toggle='modal' data-target='#pet-1'>
        <span class='img'>
          <%if (teacher.Gender == "F"){%
            <img src='images/woman icon.png', alt='icon'%>
          %}>else{%
            <img width='200px' height='320px' src='images/man.png' alt='icon'%>
          %}>
        </span>
        <span class='cover'><span class='more'>جز &rarr;</span></span>
        <span class="title"> <%=teacher.Fname + " " + teacher.Lname %> </span></a>
        <span class="details"></span>
      </div>

      <!-- Modal Start -->
      <div class="modal fade" id="pet-1" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
        <div class="modal-dialog" role="document">
          <div class="modal-content">

            <div class="modal-header">
              <h5 class="modal-title text-center" id="exampleModalLabel">عن استاذ</h5>
              <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                <span aria-hidden="true">&times;</span>
              </button>
            </div>

            <div class="modal-body">
              <%teacher.Fname + " " + teacher.Lname %>
              <p class="s "> اسم استاذ | <%=teacher.Fname + " " + teacher.Lname %></p>
              <p class="s "> <%=teacher.OfficeNumber %> | رقم المكتب</p>
              <p class="s "> <%=teacher.Email%> | البريد الإلكتروني</p>
              <p id="copy" class="ss"></p>
              <div class="row section featured topspace">
                <h5 class="section-title"><span>احجز موعدك</span></h5>
                <input id="schedule" type="button" value="جول (أيام)" onclick="window.location.href=\<%=teacher.Schedule%>;" style="position:relative; margin-left:110px;">
                <input id="back" type="button" value="احجز موعدك" style="position:relative; margin-left:90px;" onclick="window.location.href=\\"booking.html\\\";"/>
              </div>
            </div>
          </div>
        </div>
      <%});%>
    </div> <!-- /section (row section recentworks topspace)(end)-->
  </div> <!-- /container (end)-->
</main>
```

The screenshot shows a web application interface. At the top, there's a header with three dots. Below it is a main content area with a dark background. On the left, there's a sidebar with a list of names and icons. The main content area displays a grid of six teacher profiles, each with a thumbnail, name, and gender indicator. A modal dialog box is open in the center, containing fields for name, office number, email, and a copy button. Below the modal, there are two buttons labeled 'جول (أيام)' and 'احجز موعدك'. The overall design is clean and modern.

Figure 11. HTML for main page.



## 2.2.4 Booking Page

```
<main id="main">
  <form id="booking" action="/booking" method="post" class="container">

    <div class="text-right">
      <h5>اليوم:</h5>
      <select id= "Days" name="day">
        <option value="الاثنين">الاثنين</option>
        <option value="الثلاثاء">الثلاثاء</option>
        <option value="الأربعاء">الأربعاء</option>
        <option value="الخميس">الخميس</option>
      </select>

      <h5>الوقت:</h5>
      <select id= "time" name="time">
        <option value="12:00-12:30">12:00-12:30</option>
        <option value="12:30-13:20">12:30-13:20</option>
        <option value="11:00-11:10">11:00-11:10</option>
      </select>

    </div>

    <div class="text-right">
      <h5>رمز المقرر:</h5>
      <select id= "courseCode" name="code">
        <option value="CS-242">CS-242</option>
        <option value="CS-346">CS-346</option>
        <option value="CS-346">CS-467</option>
      </select>

      <h5>اللشنعة:</h5>
      <select id= "Section" name="Section">
        <option value="371">371</option>
        <option value="372">372</option>
        <option value="372">374</option>
      </select>
    </div>

    <div id="submitInfo" class="positioning" style="display: block; left: -14%; top:240px;">
      <input id="back" type="button" value="رجوع"
        onclick="window.location.href='index.html';">
      <input id="book" type="submit" value="จอง">
    </div>
  </form>
</main>
```

رمز المقرر:

الاثنين

اللشنعة:

371

الوقت:

12:00-12:30

رجوع

จอง

Figure 12. HTML for booking page.



## 2.2.5 My Booking Page

```
mybooking.html

<main id="main">
  <div class='container'>
    <table class="table" style= "background-color: white;">
      <thead >
        <tr class="text-center" style="background-color: #ECEBEB ">
          <th>هل تزيد إلغاء الحجز؟</th>
          <th>الوقت</th>
          <th>اليوم</th>
          <th>اسم الاستاذ</th>
        </tr>
      </thead>

      <tbody class="text-center">
        <%reserv_list.forEach(reserv => {%
          <tr><td><a href="/<%=reserv._id%>?_method=delete" style="color:#6F6F70;text-decoration:none; font-size: medium;">إلغاء</a></td>
          <td><%=reserv.time%></td>
          <td><%=reserv.day%></td>
          <td ><%=teacher_id_names[reserv.teacher_id]%></td>
        </tr>
        <%});%>
      </tbody>
    </div>
  </table>
</main>
```

هل تزيد إلغاء الحجز؟	الوقت	اليوم	اسم الاستاذ
----------------------	-------	-------	-------------

Figure 13. HTML for my booking page.



## 2.2.6 My Information Page

```

myinfo.html

<main id="main">

<input class="positioning" id="register" type="button" value="إحجز" onclick="window.location.href='/index';" style="left: 88%; top: -180px;" >
<form style="width:850px; height:530px; left:1%; method="post" class="container box positioning" action="/sign_up">

<div>
<div class="positioning" style="left:-10%;" >

<h5 class="textIn">الاسم الأول:</h5>
<input id= "name1" type="text" name="Fname" >

<h5 class="textIn">الاسم الأب:</h5>
<input id= "name2" type="text" name="Sname" >

<h5 class="textIn">الجنس:</h5>
<input id= "name3" type="text" name="Tname" >

<h5 class="textIn">البريد الجامعي:</h5>
<input id= "Email" type="email" name="email" >

<h5 class="textIn">كلمة المرور:</h5>
<input id= "password#1" type="password" name="password1">

<div id="submitInfo" class="positioning" style="display: block; left: -24%; bottom: -50px;">
<input id="submit" type="submit" value="احجز" name="submit1" style="width: 130px;">
</div>
</div>
</div>
</form>
</main>

```

البريد الجامعي:	الاسم الأول:
<input type="text"/>	<input type="text"/>
كلمة المرور:	اسم الأب:
<input type="text"/>	<input type="text"/>
جنس:	
الجنس:	اسم العائلة:
<input type="text"/>	<input type="text"/>

Figure 14. HTML for my information page.

## 2.2.7 About Page

```

about.html

<main id="main">
<div class="container">
<div class="row topspace">
<article > <!-- Article main content -->

<h4 style=" text-align:right-side;"><b>عن ساعة مكتبة:</b></h4>
<h5>
ساعة مكتبة هي مبادرة لقسم علوم الحاسوب في جامعة الإمام محمد بن سعود الإسلامية تهدف إلى تسهيل التواصل وربط الطالب بasta حيث يمكن الطالب حجز موعد في الساعات
المكتبية للأستاذ، وأيضاً يمكنه الوصول إلى معلومات أستاذة الأكاديمية(رقم المكتب، الجدول ، البريد الجامعي)
</h5>
<br> <br> <br><br><br><br>

</article> <!-- /Article -->
</div>
</div> <!-- /container -->
</main>

```

عن ساعة مكتبة:  
ساعة مكتبة هي مبادرة لقسم علوم الحاسوب في جامعة الإمام محمد بن سعود الإسلامية تهدف إلى تسهيل التواصل وربط الطالب بasta حيث يمكن الطالب حجز موعد في الساعات  
المكتبية للأستاذ، وأيضاً يمكنه الوصول إلى معلومات أستاذة الأكاديمية(رقم المكتب، الجدول ، البريد الجامعي)

Figure 15. HTML for about page.



### 3. Back End

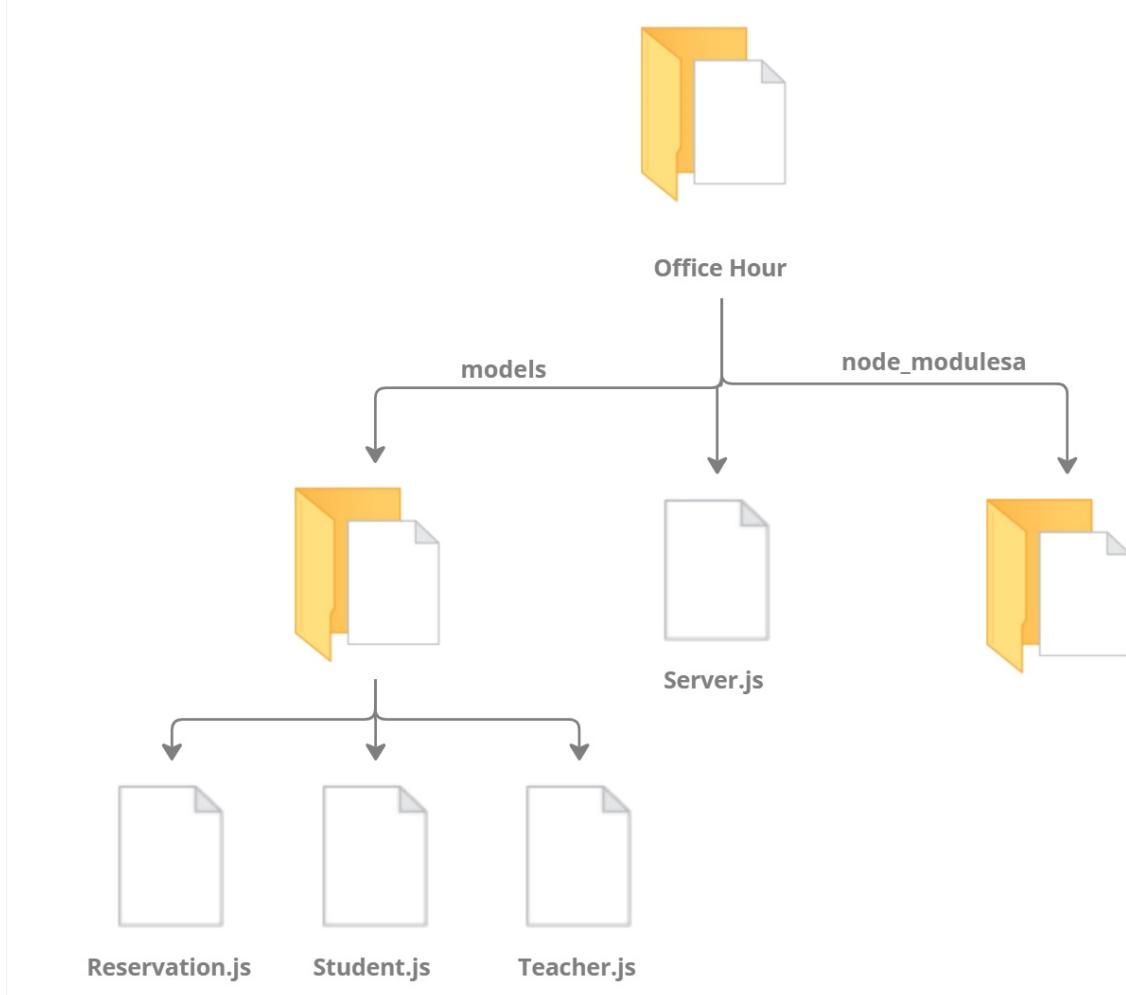


Figure 16. Back end file structure.



## 3.1 Implementation

### 3.1.1 Models Folder and Main File

- Teacher model

```
Server.js

import mongoose from 'mongoose';
const Schema = mongoose.Schema;

{
  user: {
    Fname: {
      type: String,
      required: [true, "الرجاء إدخال الاسم الأول "],
    },
    Sname: {
      type: String,
      required: [true, "الرجاء إدخال الاسم الثاني "],
    },
    Tname: {
      type: String,
      required: [true, "الرجاء إدخال الاسم الثالث "],
    },
    Lname: {
      type: String,
      required: [true, "الرجاء إدخال اسم العائلة "],
    },
    email: {
      type: String,
      required: [true, "الرجاء إدخال البريد الإلكتروني "],
    },
    Gender: {
      type: String,
      required: [true, "الرجاء إدخال الجنس "],
    },
    OfficeNumber: {
      type: String,
      required: [true, "الرجاء إدخال رقم المكتب "],
    },
    Schedule: {
      type: String,
      required: [true, "الرجاء إدخال رابط الجدول "],
    },
    Office_Hours: {
      Day_Time: {
        type: Object,
        required: [true, "الرجاء إدخال اليوم والوقت لساعة المكتبة "],
      }
    },
    Course: {
      Subject_Section: {
        type: Object,
        required: [true, "الرجاء إدخال المادة والشعبة "],
      }
    },
    {
      timestamps: true
    }
  };
}

const TeacherSchema = mongoose.model("Teacher", Schema);

export default TeacherSchema;
```

We need a schema in order to insert our data into the database. Schema helps us to ensure there are no unintended schema changes or improper data types. The teacher schema is shown in Figure 17.

Figure 17. Data schema for teacher.



- Student model

```
Server.js

import mongoose from 'mongoose';
const Schema = mongoose.Schema();

{
  user: {
    Fname: {
      type: String,
      required: [true, "الرجاء إدخال الاسم الأول"],
    },
    Sname: {
      type: String,
      required: [true, "الرجاء إدخال الاسم الثاني"],
    },
    Tname: {
      type: String,
      required: [true, "الرجاء إدخال الاسم الثالث"],
    },
    Lname: {
      type: String,
      required: [true, "الرجاء إدخال اسم العائلة"],
    },
    email: {
      type: String,
      required: [true, "الرجاء إدخال البريد الإلكتروني"],
    }
  },
  password: {
    type: String,
    required: [true, "الرجاء إدخال كلمة السر"],
  },
  ReservationList: {
    type: Array,
    required: false,
  }
},
{
  timestamps: true
};

const StudentSchema = mongoose.model("Student", Schema);

// export const schema = Student.schema;
export default StudentSchema;
// module.exports = Student;
```

We need a schema in order to insert our data into the database. Schema helps us to ensure there are no unintended schema changes or improper data types. The student schema is shown in Figure 18.

Figure 18. Data schema for student.



- Reservation model

```
Server.js

import mongoose from 'mongoose';
const Schema = mongoose.Schema;

{
  day: {
    type: String,
    required: [true, "الرجاء إدخال الاسم الأول"],
  },
  time: {
    type: String,
    required: [true, "الرجاء إدخال الاسم الثاني"],
  },
  code: {
    type: String,
    required: [true, "الرجاء إدخال الاسم الثالث"],
  },
  Section: {
    type: String,
    required: [true, "الرجاء إدخال اسم العائلة"],
  },
  Status: {
    type: String,
    required: [true, "الرجاء إدخال البريد الإلكتروني"],
  },
  teacher_id: {
    type: String,
    required: [true, "الرجاء إدخال معرف المعلم"],
  },
  student_id: {
    type: String,
    required: [true, "الرجاء إدخال معرف الطالب"],
  }
},
{
  timestamps: true
};

const ReservationSchema = mongoose.model("Reservation", Schema);

// export const schema = Reservation.schema;
export default ReservationSchema;
//module.exports = Reservation;
```

We need a schema in order to insert our data into the database. Schema helps us to ensure there are no unintended schema changes or improper data types. The reservation schema is shown in Figure 19.

Figure 19. Data schema for reservation.



- Classes in main.js

```
Server.js

class User{
    constructor(Fname, Sname, Tname, Lname, Email) {
        this.Fname = Fname;
        this.Sname = Sname;
        this.Tname = Tname;
        this.Lname = Lname;
        this.Email = Email;
    }
}

class Reservation{
    constructor(date, Day, Time, Section, Coursecode, Teacher_id, Student_id) {
        this.Status = "Active";
        this.Day = Day;
        this.Time = date.getHours() + ":" + date.getMinutes() + ":" + date.getSeconds() + ":" +
        date.getMilliseconds();
        this.date = date;
        this.Section = Section;
        this.Course_code = Coursecode;
        this.Teacher_id = Teacher_id;
        this.Student_id = Student_id;
    }
}

class Student extends User{
    constructor(Fname, Sname, Tname, Lname, Email, Password) {
        super(Fname, Sname, Tname, Lname, Email);
        this.Password = Password;
        this.Reservation_list = new Array();
    }
}

export class Teacher extends User{
    constructor(Fname, Sname, Tname, Lname, OfficeNumber, Email, Gender, Schedule, ID) {

        super(Fname, Sname, Tname, Lname, Email);
        this.Office_Hours = null;
        this.Schedule = Schedule;
        this.OfficeNumber = OfficeNumber;
        this.Gender = Gender;
        //this.full_name = Arabic_name;
        this.ID = ID;
    }
}
```

Figure 20. Teacher, Student and Reservation classes.

To deal with our data after we retrieve them from the database, we have to create classes to create objects and store the retrieved data. See Figure 20.



### 3.1.2 Server File

- Initialization

```
Server.js

// import teacher class to create objects.
import {Teacher} from './assets/js/Main.js';

// use it for send emiles to the teacher.
import {createTransport} from 'nodemailer';
import mailgen from 'mailgen';

// to process connection request and response.
import express from 'express';

// to help us with mongodb
import mongoose from 'mongoose';
import morgan from 'morgan';

// used to parse incoming bodies.
import bodyParser from 'body-parser';

// import models
import StudentSchema from './models/Student.js';
import ReservationSchema from './models/Reservation.js';
import TeacherSchema from './models/Teacher.js';

// help us to find file path
import path from 'path';
import { fileURLToPath } from 'url';
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);

// connect to mongoDB (start)=====
mongoose.connect('mongodb://127.0.0.1:27017/test', { useNewUrlParser: true,
useUnifiedTopology: true });

const db = mongoose.connection;

db.on("error", console.error.bind(console, "connection error: "));

db.once("open", function () { console.log("Connected successfully");});

// connect to mongoDB (end)=====

// create app (start) =====

const app = express();

// to read body of the request
app.use(morgan('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

// specificie the port
let port = 8080;

// determen static and dynamic folders
app.use(express.static(path.join(__dirname, 'assets')));
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

// create app (end) =====
```



In Figure 21, initially, we import all necessary packages, methods, classes, and modules. Then we connect to the database. Lastly, we create an app and set up some tools that help to encode the body request and determine the static and

Figure 21. initialization for Server.js



- The route to the sign in page.

```
Server.js

app.get('/', async function(req, res) {
  res.sendFile(path.join(__dirname, '/assets/signin.html'));
});
```

Figure 22. the route for sin in page.

- The route to the main page.

```
Server.js

app.get('/index', async function(req, res) {
  try{
    const teacher = await TeacherSchema.find({}).then( async function(teacher_info){
      const number_of_teacher = teacher_info.length;

      let teachers_list = [];

      for(let i=0; i< number_of_teacher;i++){

        let Fname = teacher_info[i].user.Fname;
        let Sname = teacher_info[i].user.Sname;
        let Tname = teacher_info[i].user.Tname;
        let Lname = teacher_info[i].user.Lname;
        let Office_numbe = teacher_info[i].OfficeNumber;
        let Email = teacher_info[i].user_email;
        let Gender = teacher_info[i].Gender;
        let Schedule = teacher_info[i].Schedule;
        let ID = teacher_info[i]._id;

        let teacher_obj = new Teacher(Fname, Sname, Tname, Lname, Office_numbe, Email, Gender,
        Schedule, ID );
        teachers_list.push(teacher_obj);
      }

      console.log(teachers_list)
      res.render('index', {teachers_list: teachers_list});
    })
  }).catch(function(err){
    console.log(err);
  });

} catch(error){
  console.log(error.message);
  res.status(500).json({message: error.message});
}
});
```

Figure 23. The route to the main page

When user send a get request to the server on the root “/”. Server will response by sending the sign in page to the user. As shown in Figure 22.



When a user requests "/index", the main page. To display the main page to the user, we need to retrieve the teacher information from the database, give it to index.ejs, and then illustrate it. We converted the index file to ejs format so that it can get retrieved data from the database and then store it in an array called teachers\_list and then render the index.ejs with teachers\_list. as shown in Figure 23.



- The route to my booking page.

```
Server.js

app.get('/mybooking', async function(req, res) {
try{
    ReservationSchema.find({}).then( async function(reservations){
        let ids = [];

        reservations.forEach(e => {ids.push(e.teacher_id)});

        let Ta_ids = await TeacherSchema.find({_id: { $in: Array.from(ids) }});

        let teacher_names = {}

        for (let [x, y] of zip([ids, Ta_ids])) {
            let name = y.user.Fname + " " +y.user.Lname;
            teacher_names[x] = name;
        }

        res.render('mybooking', {reserv_list: reservations, teacher_id_names: teacher_names}).catch(function(err){console.log(err);})

    }).catch(function(err){
        console.log(err);
    });
} catch(error){
    console.log(error.message);
    res.status(500).json({message: error.message});
}
});
```

الصلحة الرئيسية	عن ساعة مكتبة	محور انتي	تعديل بياناتي
هل تزيد المدة المجزأ؟	الوقت	اليوم	اسم المستلم
[إلغاء]	12:00-12:30	الإثنين	بسـهـ الصـوـلـيـ

When a user requests the “/mybooking” our website will retrieve all bookings that the user did from a database and then illustrate them on the page. The page has ejs format for the same reason as the index page. As you can see in Figure 24 we store all retrieved data in the reservations list. Additional to reservation data we have to retrieve the teacher name associated with each reservation.

Figure 24. The route to my booking page.



## • Booking post request

```

  . . .
  Server.js

app.post('/booking', async function(req, res) {
  try {
    let time = req.body.time;
    let code = req.body.code;
    let day = req.body.day;
    let Section = req.body.Section;

    let teacherID = "647a18c7f2b4d7b581b47e78";
    let studentID = "647320b808086dd71d7f804";

    let Status = "active";

    const reservation = await ReservationSchema.create({ day: day, time: time, "code": code, "Section": Section, "Status": Status, "teacher_id": teacherID, "student_id": studentID });

    let student = await StudentSchema.findById({ _id: studentID });
    console.log(student[0]);

    student = student[0];
    //const name = '';
    //let student_name = '';
    const student_full_name = student.user.Fname + " " + student.user.Sname + " " +
    student.user.Fname + " " + student.user.Sname;
    const student_email = student.user.email;

    ReservationSchema.find({}).then(async function(reservations) {
      //console.log("reservations.teacher_id: ", reservations); { results: { $elemMatch: { $gte: 80, $lt: 85 } } }
      let ids = [];

      reservations.forEach(e => ids.push(e.teacher_id));
      let Ta_ids = await TeacherSchema.find({ "_id": { $in: Array.from(ids) } });

      let teacher_names = {};

      for (let [x, y] of zip([ids], Ta_ids)) {
        let name = y.user.Fname + " " + y.user.Sname;
        teacher_names[x] = name;
      }

      for (const prop in teacher_names) {
        console.log(`key: ${prop}, value: ${teacher_names[prop]}`);
      }
    });

    const userEmail = "wksalghaith@sm.imamu.edu.sa";

    let config = {
      service: 'gmail',
      auth: {
        user: 'imam.cs.oh@gmail.com',
        pass: 'f p g c o n k e d n n i t u x i'
      }
    };

    let transporter = createTransport(config);

    let Mailgenerator = new mailgen({
      theme: 'default',
      product: {
        name: "ساعة مكتبة",
        link: "https://mailgen.js/"
      }
    });

    let response = {
      body: {
        name: '',
        intro: "أهلاً بك في حجز موعد في ساعتك المكتبة",
        table: {
          data: [
            {
              "العنوان": student_full_name,
              "الرمز": code,
              "الفصل": Section,
              "النوع": day,
              "الوقت": time,
              "العنوان": student_email,
            }
          ],
          outre: "آمنة وفعالة"
        }
      }
    };

    transporter.sendMail(message).catch(error => {
      return res.status(500).json({ error });
    })
  };
  . . .
  res.render('mybooking', { reservation_list: reservations, teacher_id_names: teacher_names }).catch(function(err){console.log(err);});

  . . .
  res.render('mybooking', { reservation_list: reservations, teacher_id_names: teacher_names }).catch(function(err){console.log(err);});

  . . .
  res.render('mybooking', { reservation_list: reservations, teacher_id_names: teacher_names }).catch(function(error){
    console.log(error.message);
    res.status(500).json({message: error.message});
  });
  . . .
}

```

When the user press “حجز” button, our app should take the information from the body request and add a new reservation to the database. A notification message will be sent to the teacher to notify him about the reservation. Lastly, the page will be updated after the addition process and show the new reservation. As shown in Figure 25.

Figure 25. Booking post request.



## ● Cancel reservation

```

*** Server.js

app.get('/:id', async function(req, res) {
  try{
    let studentID = '647520b0800866dd71d7fb04';

    let reservation_info = await ReservationSchema.find({_id : req.params.id});

    reservation_info = reservation_info[0];

    let time = reservation_info.time;
    let code = reservation_info.code;
    let day = reservation_info.day;
    let Section = reservation_info.Section;
    let teacherID = '647505a37c0e469ac27af03';

    await ReservationSchema.deleteOne({_id: req.params.id})

    ReservationSchema.findById().then( async function(reservation){

      let ids = [];
      reservations.forEach(e => {ids.push(e.teacher_id)});

      let Ta_ids = await TeacherSchema.find({_id: { $in: Array.from(ids) }});

      let teacher_names = {};

      for (let [x, y] of zip([ids, Ta_ids])) {
        let name = y.user.name + " " + y.user.name;
        teacher_names[x] = name;
      }

      for (const prop in teacher_names) {
        console.log(`key: ${prop}, value: ${teacher_names[prop]}`);
      }

      //***** 

      let student = await StudentSchema.findById({_id : studentID});
      console.log(student[0]);

      student = student[0];
      //const name = '';
      //let student_name = '';
      const student_full_name = student.user.Fname + " " + student.user.Sname + " " +
      student.user.Fname + " " + student.user.Lname;
      const student_email = student.user.email;

      const userEmail = 'wksalghait@sm.imamu.edu.sa';

      let config = {
        service: 'gmail',
        auth: {
          user: 'imam.cs.oh@gmail.com',
          pass: 'f p g c o n k s d n n l t w x i'
        }
      }

      let transporter = createTransport(config);

      let MailGenerator = new mailgen({
        theme: "default",
        product: {
          name: "نكتة",
          link: "https://mailgen.js/"
        }
      })

      let response = {
        body: {
          name : teacher_names[teacherID],
          intro: "عزيزي معلم عزيزنا",
          table: {
            data: [
              {
                "العنوان": student_full_name,
                "الرمز": code,
                "الفصل": Section,
                "النوع": day,
                "الوقت": time,
                "البريد الإلكتروني": student_email,
              }
            ],
            outro: "شكراً لك وشكراً"
          }
        }
      }

      transporter.sendMail(message).catch(error => {
        return res.status(500).json({ error })
      })
      //***** 

      res.render('canceling', {reserv_list: reservations, teacher_id_names: teacher_names}).catch(function(err){console.log(err);})
    }).catch(function(err){
      console.log(err);
    });
  } catch(error){
    console.log(error.message);
    res.status(500).json({message: error.message});
  }
});

```

When user press “إلغاء” we will search on the database by the id of this reservation and then store the retrieved object to extract the related information with that reservation such as the course, section, time. After we take the information now we will remove it from the database. In addition we will take the student name and his official email. Lastly we our app will send an email to notify the teacher about the reservation cancelation with collected information (course code, section, time, student name, student official email). As shown in Figure 26.

Figure 26. Cancel reservation route.



- Sign in to the website

```
Server.js

app.post("/signin", async function(req, res){
  try {
    const email = req.body.username;
    console.log(email);
    // check if the user exists
    const user = await StudentSchema.findOne({ 'user.email':
      "wksalghait@sm.imamu.edu.sa"});

    if (user) {
      //check if password matches
      const result = req.body.password === user.password;
      if (result) {
        let teachers_list = [];

        const teacher = await TeacherSchema.find({}).then( async function(teacher_info){
          const number_of_teacher = teacher_info.length;

          for(let i=0; i < number_of_teacher;i++){
            let Fname = teacher_info[i].user.Fname;
            let Sname = teacher_info[i].user.Sname;
            let Tname = teacher_info[i].user.Tname;
            let Lname = teacher_info[i].user.Lname;
            let Office_numbe = teacher_info[i].OfficeNumber;
            let Email = teacher_info[i].user.email;
            let Gender = teacher_info[i].Gender;
            let Schedule = teacher_info[i].Schedule;
            let ID = teacher_info[i]._id;

            let teacher_obj = new Teacher(Fname, Sname, Tname, Lname, Office_numbe, Email,
              Gender, Schedule, ID );
            teachers_list.push(teacher_obj);
          }
        });

        //console.log(teachers_list)
        res.render('index', {teachers_list: teachers_list});

        } else {
          res.status(400).json({ error: "password doesn't match" });
        }
      } else {
        res.status(400).json({ error: "User doesn't exist" });
      }
    } catch (error) {
      console.log(error);
      res.status(400).json({ error });
    }
  };
});
```

The form has two input fields: 'البريد الجامعي:' (Official Email) containing 'mosalomar@sm.imamu.edu.sa' and 'كلمة المرور:' (Password) which is empty. Below the fields is a large blue button labeled 'دخول' (Login) with a white cursor arrow pointing to its right side.

When the user presses the “دخول” the website will take the official email and password from the body request and then search on the database if there is any student who registered with the entered email. If does not exist, it will print an error message “{error: User doesn't exist}”. If the user exists, then we will retrieve his stored password and compare it with the entered one if matches the website will allow to user to enter, and it will give it the main page. If the password does not match, the website will present the “{error: password doesn't match}”. As shown in Figure 27.

Figure 27. Sign in post request.



- Sign up to the website

```
Server.js

app.post("/sign_up", async function(req, res){

try{
    console.log("Body ", req.body);
    let Fname = req.body.Fname;
    let Sname = req.body.Sname;
    let Tname = req.body.Tname;
    let Lname = req.body.Lname;
    let email = req.body.email;
    let password1 = req.body.password1;
    let password2 = req.body.password2;

    const student = await StudentSchema.create({ "user": {
        "Fname": Fname,
        "Sname": Sname,
        "Tname": Tname,
        "Lname": Lname,
        "email": email
    },
    "password": password1});

    let teachers_list = [];

    const teacher = await TeacherSchema.find({}).then( async function(teacher_info){
        const number_of_teacher = teacher_info.length;

        for(let i=0; i< number_of_teacher;i++){
            let Fname = teacher_info[i].user.Fname;
            let Sname = teacher_info[i].user.Sname;
            let Tname = teacher_info[i].user.Tname;
            let Lname = teacher_info[i].user.Lname;
            let Office_numbe = teacher_info[i].OfficeNumber;
            let Email = teacher_info[i].user.email;
            let Gender = teacher_info[i].Gender;
            let Schedule = teacher_info[i].Schedule;
            let ID = teacher_info[i]._id;

            let teacher_obj = new Teacher(Fname, Sname, Tname, Lname, Office_numbe, Email, Gender,
            Schedule, ID );
            teachers_list.push(teacher_obj);
        }
    })

    res.render('index', {teachers_list: teachers_list});

} catch(error){
    console.log(error.message);
    res.status(500).json({message: error.message});
}
});
```

When the user presses the “تسجيل” button, our website will take the information with the body request and add it to the database. After that, our website will let the user enter the site and render for him the main page. As shown in Figure 28.

Figure 28. Sign up post request.