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	440023306
Khloud Alnufaie	khaalnufaie@sm.imamu.edu.sa	440020617
Raghad Albosais	rkaalbosais@sm.imamu.edu.sa	440020209
Alhanouf Almansour	aalmansour83@sm.imamu.edu.sa	440019183

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
2.1.1 Signup Function 6
2.1.2 Sign-in Function 7
2.1.3 Booking Function 7
2.1.4 Display Teacher's Information 8
2.1. 5 Cancel Reservation 10
..... 11
2.1.6 Modify Information Function 11
..... 11
..... 11
..... 11
2.2 Implementation 12
2.2.1 General structure for all pages 13
2.2.2 Sign up Page 14
22.2 Sign in Page. 14
2.2.3 Main Page 15
2.2.4 Booking Page 16
2.2.5 My Booking Page 17
2.2.6 My Information Page 18
2.2.7 About Page 18
3. Back End 19
..... 19
3.1 Implementation 20
3.1.1 Models Folder and Main File 20
3.1.2 Server File 24

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.

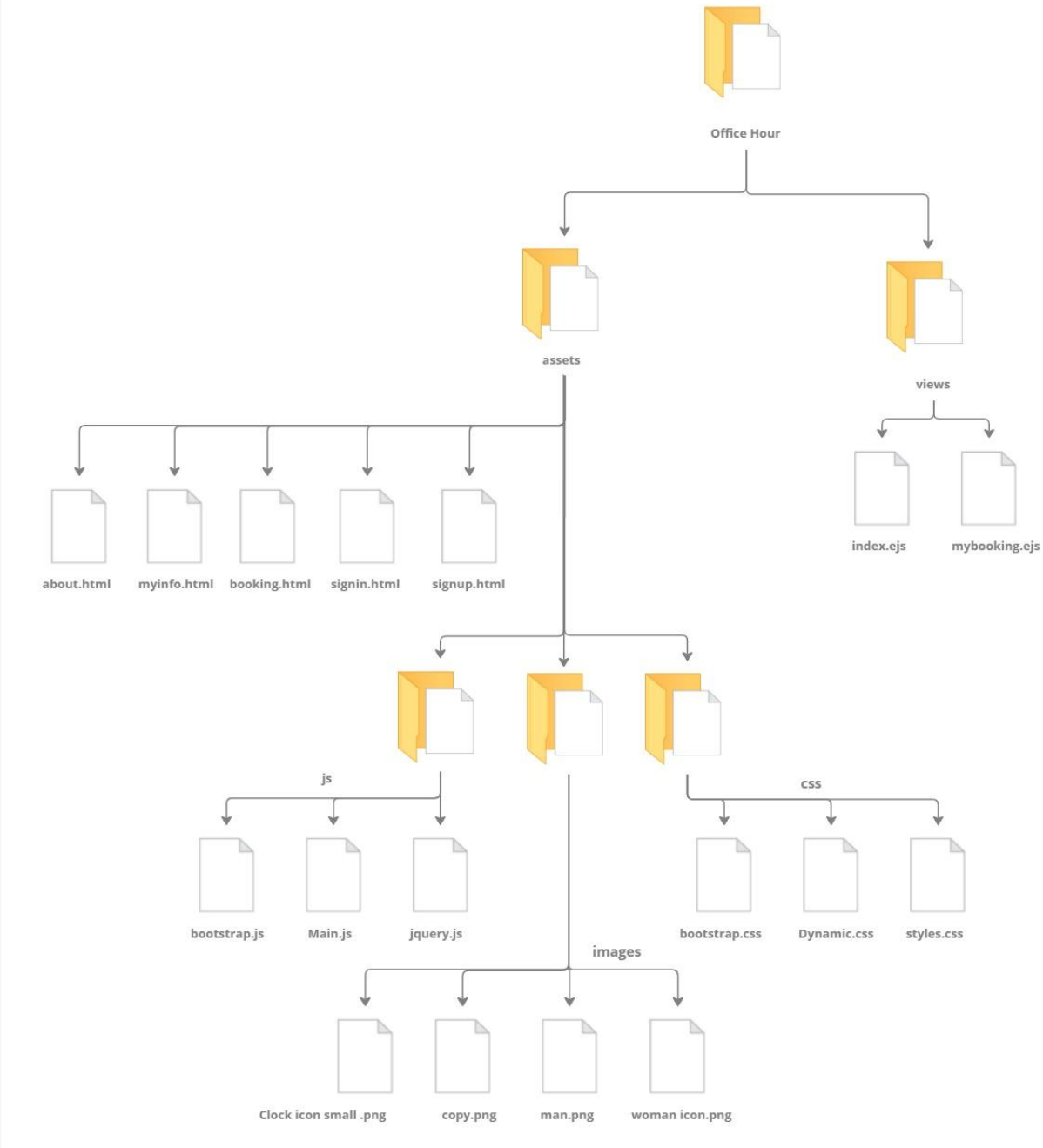
1.2 Aims

- 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


Office Hour
assets
views
about.html
myinfo.html
booking.html
signin.html
signup.html
index.ejs
mybooking.ejs
js
css
bootstrap.js
Main.js
jquery.js
bootstrap.css
Dynamic.css
styles.css
images
Clock icon small .png
copy.png
man.png
woman icon.png

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

Figure 1. Sign up function.

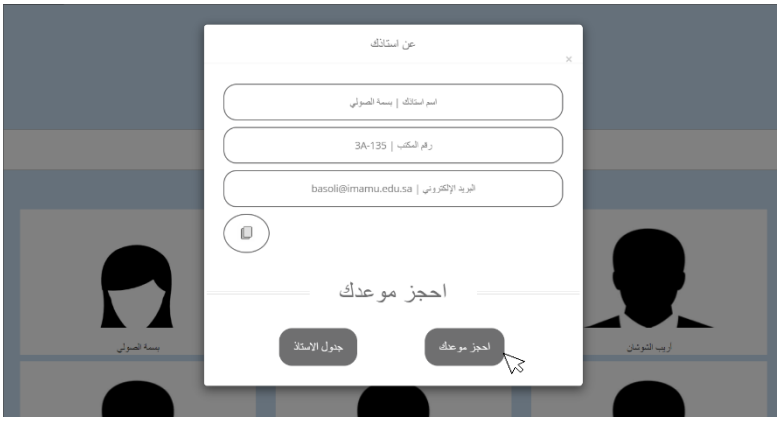
Step 1: The user will Click on "تسجيل جديد" button.
Step 2: The user will enter (First name, Second name, Third name, Fourth name,
Official email, password, confirm password).
Step 3: The user clicks "تسجيل" button.

2.1.2 Sign-in Function

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



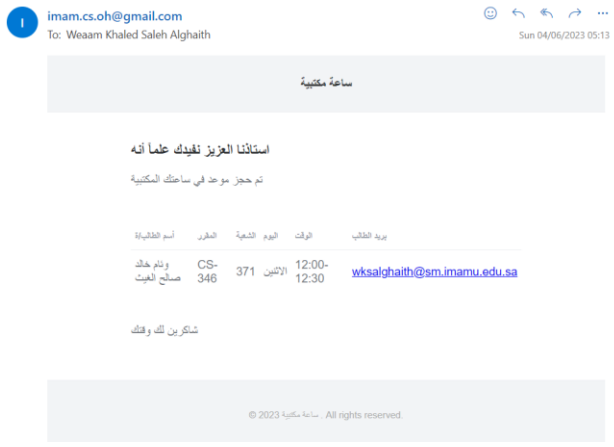
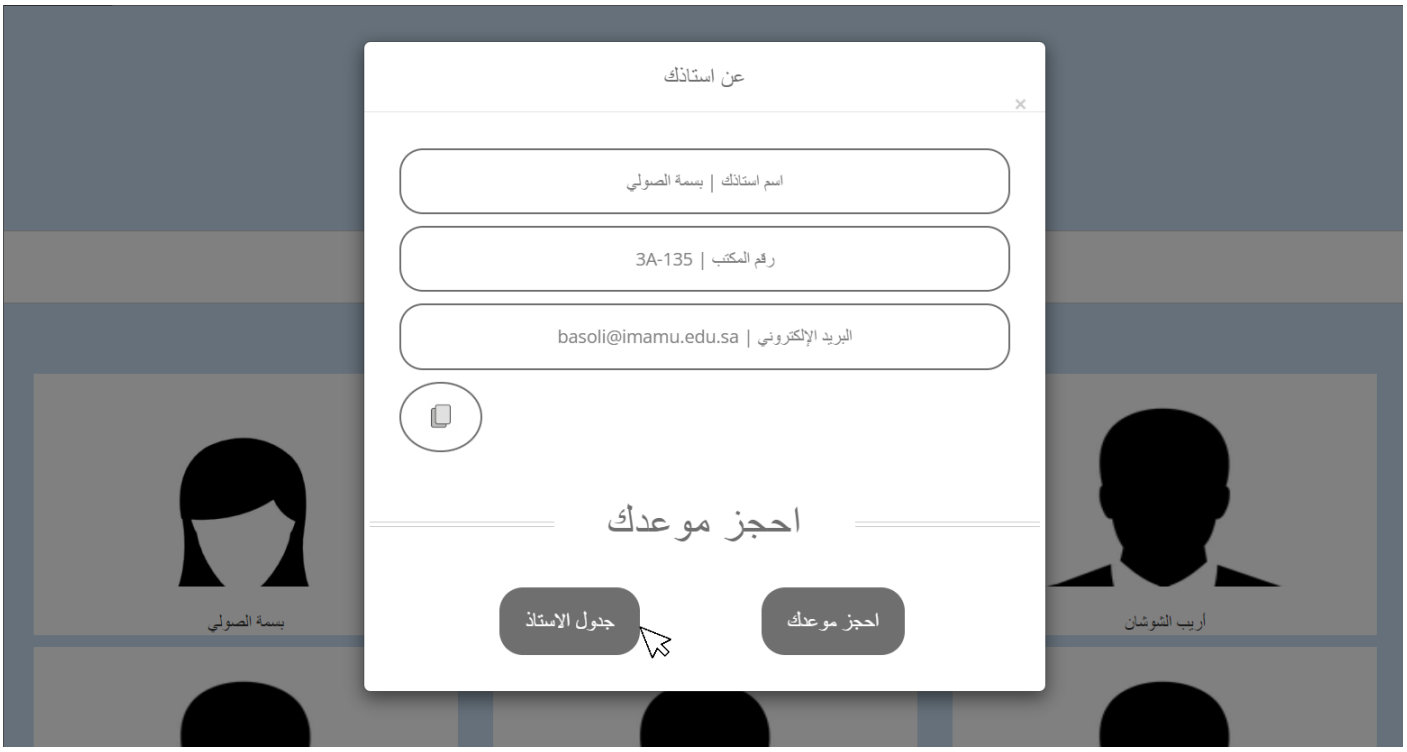


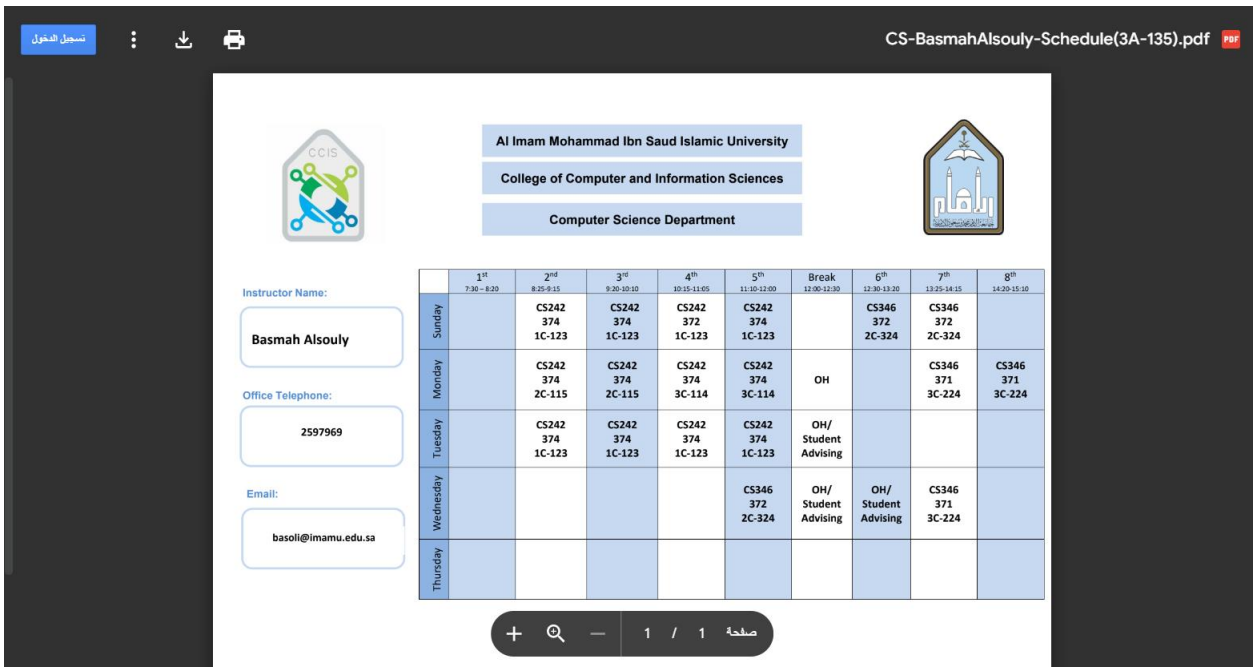
Figure 3. Booking function.

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





	1st 7:30 – 8:20	2nd 8:25-9:15	3rd 9:20-10:10	4th 10:15-11:05	5th 11:10-12:00	Break 12:00-12:30	6th 12:30-13:20	7th 13:25-14:15	8th 14:20-15:10
Sunday		CS242 374 1C-123	CS242 374 1C-123	CS242 372 1C-123	CS242 374 1C-123		CS346 372 2C-324	CS346 372 2C-324	
Monday		CS242 374 2C-115	CS242 374 2C-115	CS242 374 3C-114	CS242 374 3C-114	OH		CS346 371 3C-224	CS346 371 3C-224
Tuesday		CS242 374 1C-123	CS242 374 1C-123	CS242 374 1C-123	CS242 374 1C-123	OH/ Student Advising			
Wednesday					CS346 372 2C-324	OH/ Student Advising	OH/ Student Advising	CS346 371 3C-224	
Thursday									

Figure 4. Display teacher information.

2.1. 5 Cancel Reservation

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.

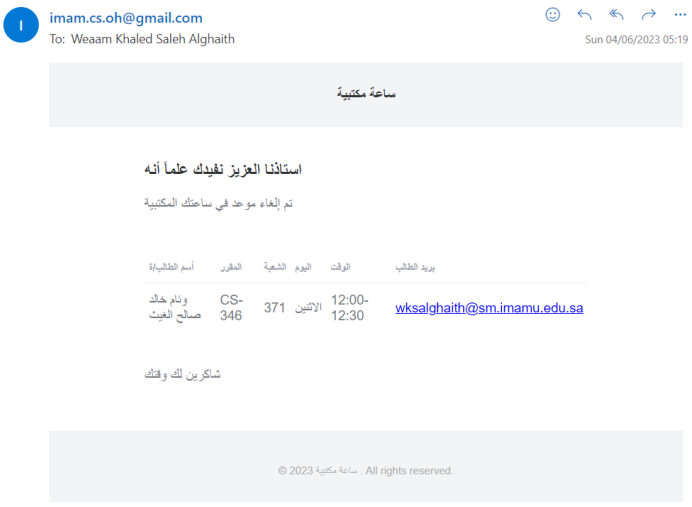


Figure 5. Cancel reservation function.

2.1.6 Modify Information Function

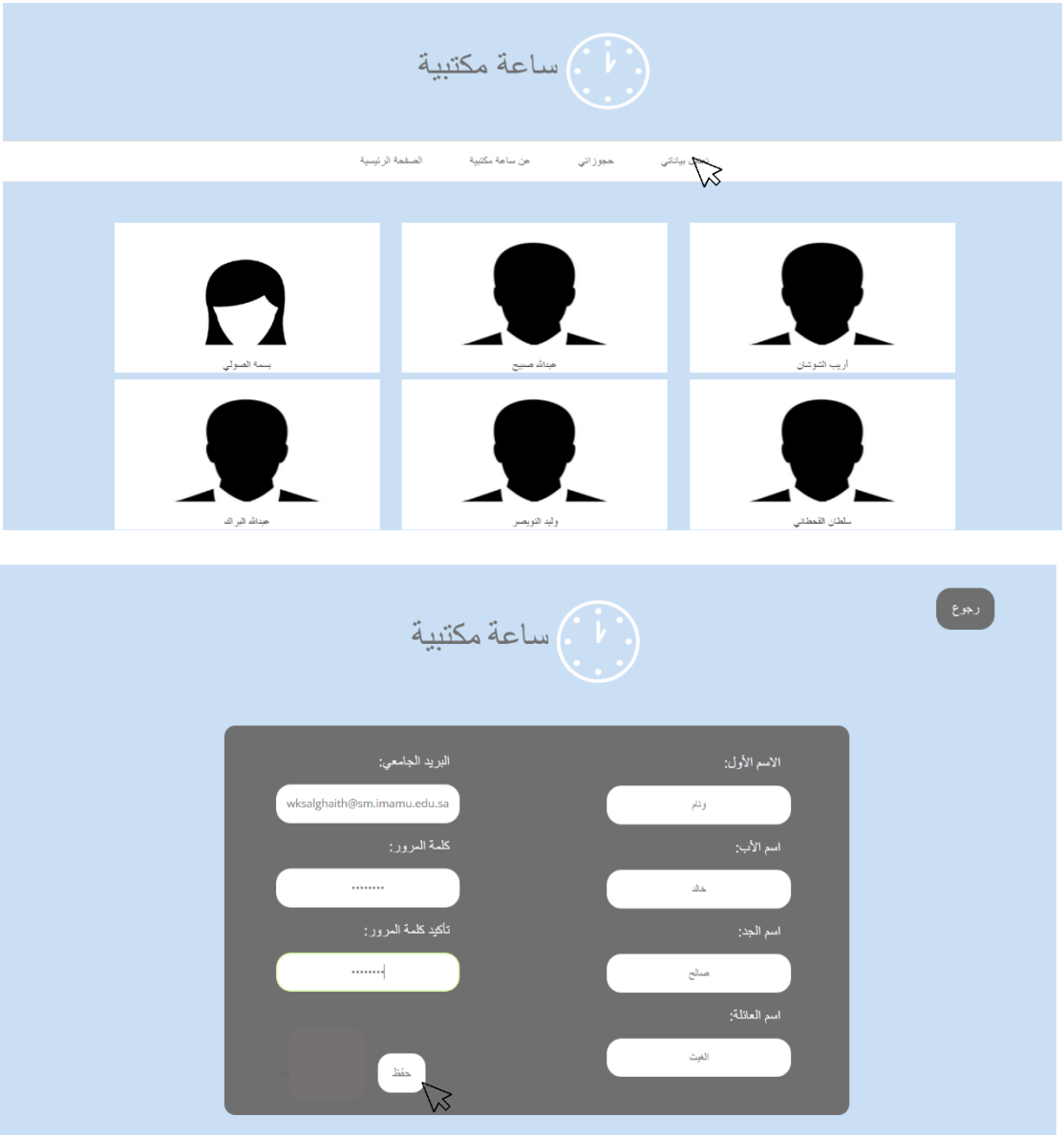


Figure 6. Modify my information.

Step 1: The user will click on "تعديل بياناتي " 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

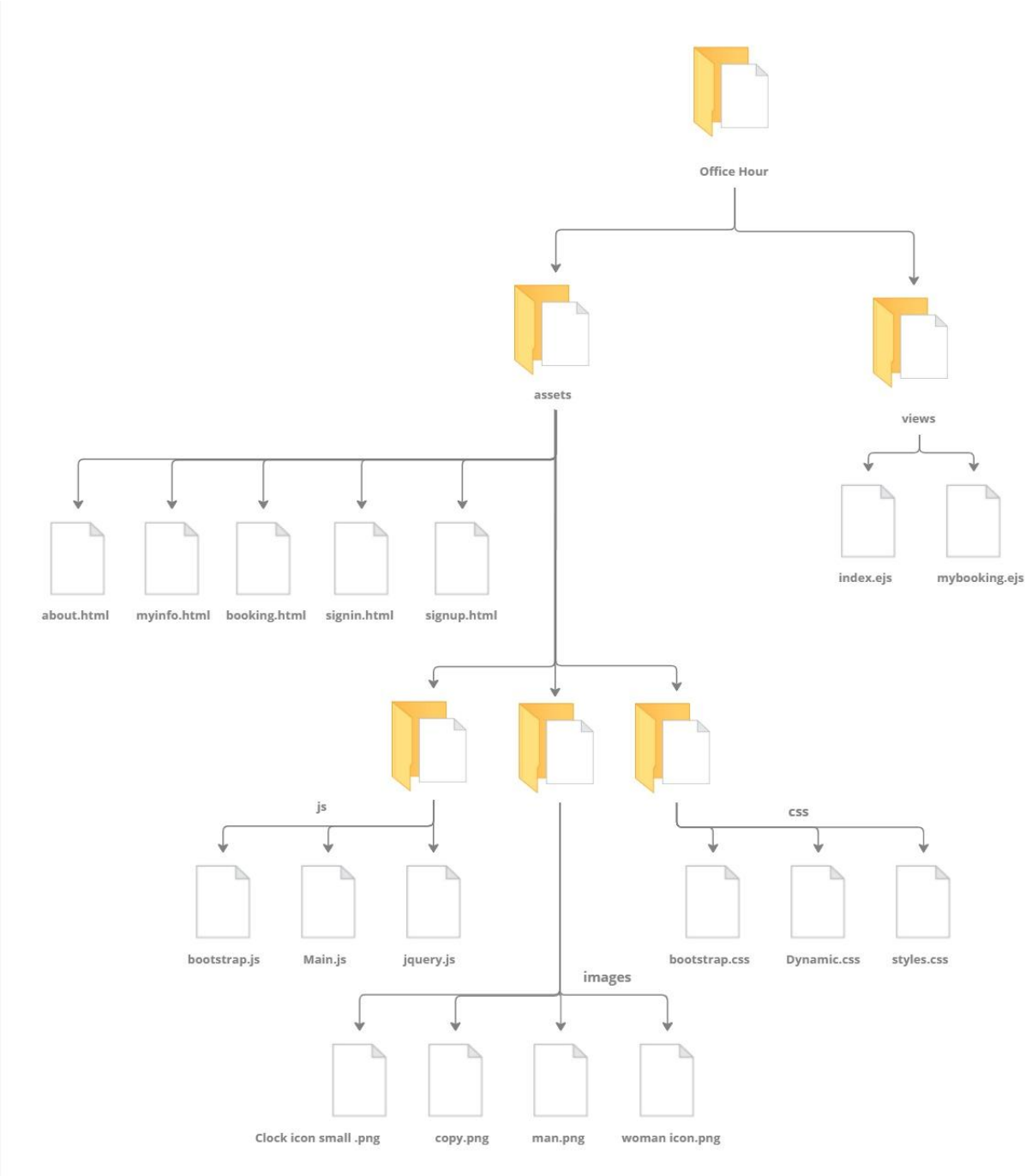


Figure 7. Front end files structure.

2.2.1 General structure for all pages

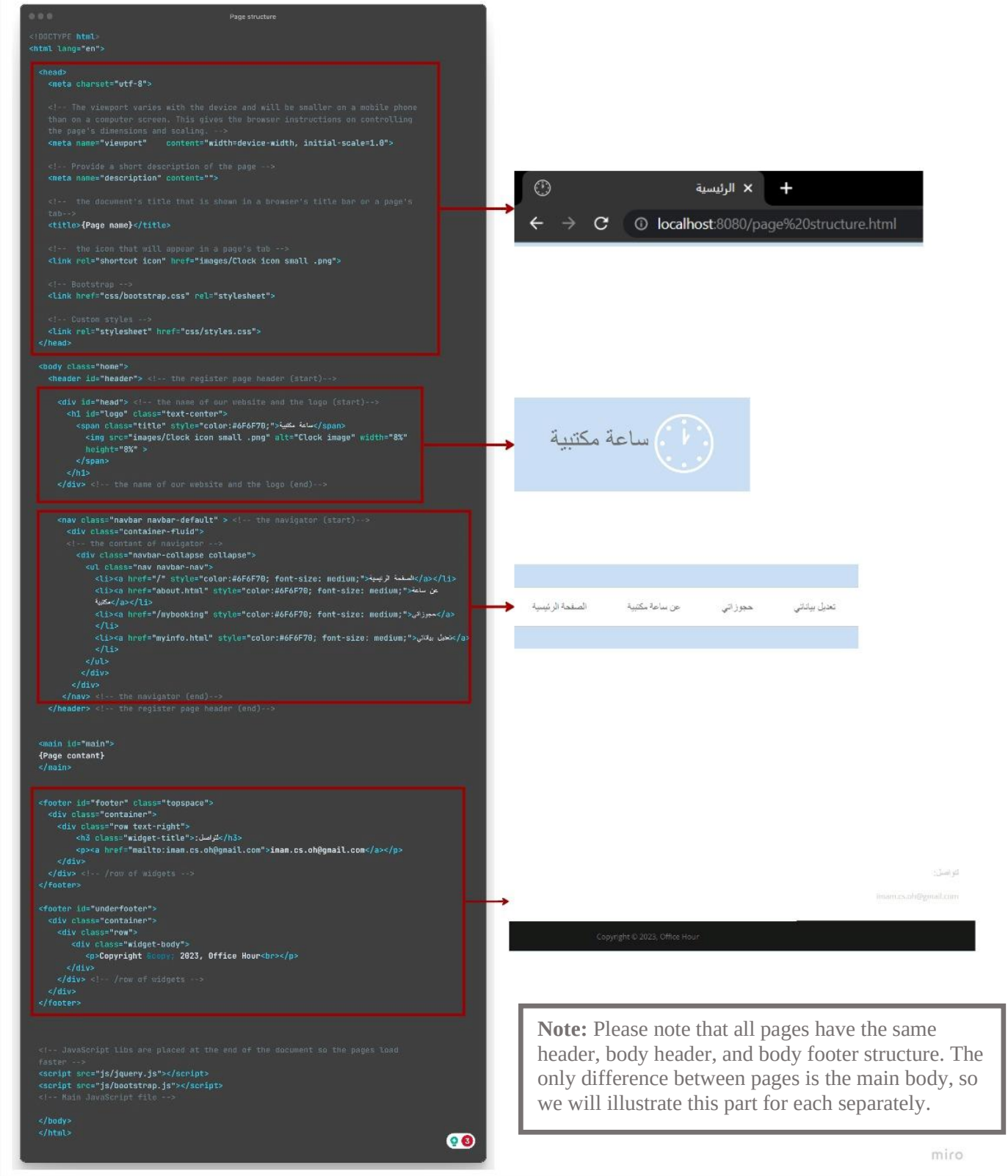


Figure 8. General HTML structure for all pages.

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="مها">

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

         <h5 class="textIn">:اسم الجد</h5>
         <input id= "name3" type="text" name="Tname" placeholder="سليمان">

         <h5 class="textIn">:اسم العائلة</h5>
         <input id= "name4" type="text" name="Lname" placeholder="العمر">

       </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>
```

Figure 9. HTML for sign up.

22.2 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;">
          <input id="submit" type="submit" name="submit" value="دخول" >
        </div>
      </div>
    </form>
</main>
```

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 class='cover'><span class='more'>حجز &rarr;</span></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 استاذك</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.Lsname %>
                  <p class="s ">  اسم استاذك | <%=teacher.Fname +" "+ teacher.Lsname %></p>
                  <p class="s"> <%=teacher.OfficeNumber %> | رقم المكتب</p>
                  <p class="s "> <%=teacher.Email%> | البريد الإلكتروني</p>
                  <p id= "copy" class="ss"><img src="images/copy.png", alt="icon"
                   onclick="copy()" ></p>
                    <div class="row section featured topspace">
                      <h5 class="section-title"><span>احجز موعدك</span></h5>
                      <input  id="schedule" type="button" value="جدول الاستاذ"
                      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>
        </div>
      </div>
      <%});%>
    </div> <!-- /section (row section recentworks topspace)(end)-->
  </div>  <!-- /container (end)-->
</main>
```

Figure 11. HTML for main page.

2.2.4 Booking Page

booking.html

```
<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>
      </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>:الشعبة</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="حجز">
    </div>

    </form>
</main>
```

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 تريد 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>
```

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="رجوع"
  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">:اسم الأب</h5>
      <input id= "name2" type="text" name="Sname">

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

      <h5 class="textIn">:اسم العائلة</h5>
      <input id= "name4" type="text" name="Lname" >
    </div>

    <div class="positioning" style="right:100%; bottom: 480px">

      <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="حفظ" name="submit1" style="width: 130px;">
      </div>
    </div>
  </div>
 </form>
</main>
```

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 مكتبية:</b></h4>
      <h5>
      ساعة مكتبية هي مبادرة لقسم علوم الحاسب في جامعة الامام محمد بن سعود الإسلامية تهدف إلى تسهيل التواصل وربط الطالب باستاذه حيث
      يستطيع الطالب حجز موعد في الساعات
      المكتبية للأستاذ، وأيضاً يمكنة الوصول إلى معلومات أستاذه الأساسية(رقم المكتب، الجدول ، البريد الجامعي)
      </h5>
      <br> <br> <br><br><br><br>

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

عن ساعة مكتبية:

ساعة مكتبية هي مبادرة لقسم علوم الحاسب في جامعة الامام محمد بن سعود الإسلامية تهدف إلى تسهيل التواصل وربط الطالب باستاذه حيث يستطيع الطالب حجز موعد في الساعات المكتبية للأستاذ، وأيضاً يمكنة الوصول إلى معلومات أستاذه الأساسية(رقم المكتب، الجدول ، البريد الجامعي)

Figure 15. HTML for about page.

3. Back End

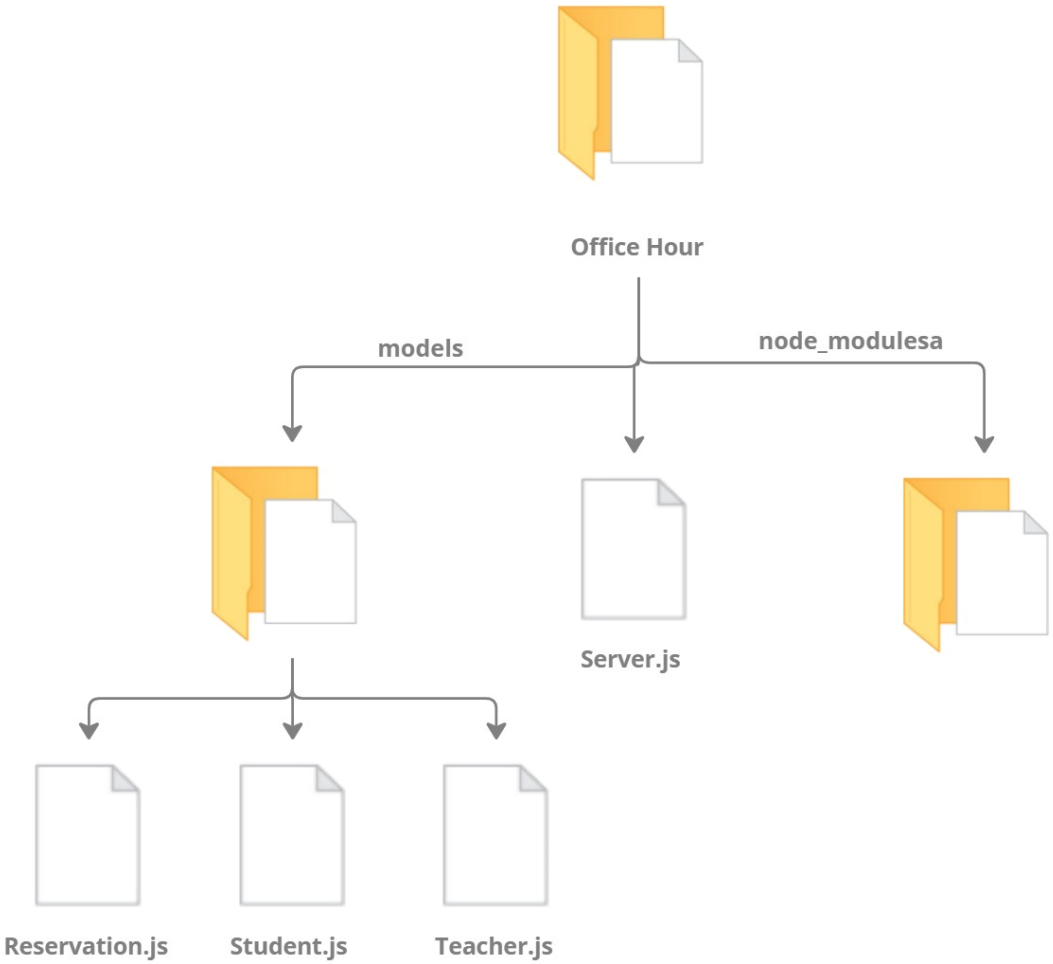


Figure 16. Back end file structure.

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 ادخال 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 ادخال اليوم والوقت لساعة المكتبية"],
    }
  },
  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, Courseccode, 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 = Course_code;
  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 mangoDB (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 mangoDB (end)===============================================


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

const app = express();

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

// specifie 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 determent 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'));
});
```

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.

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});
  }
});
```

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.

Figure 23. The route to the main page

- The route to my booking page.

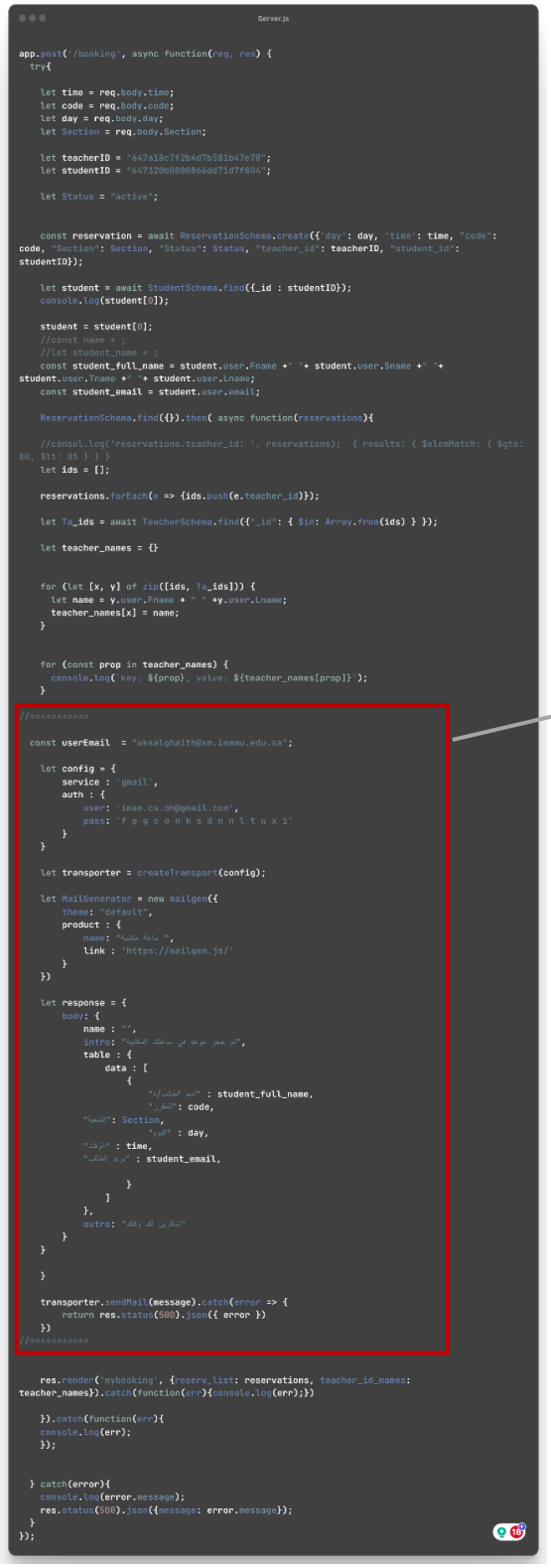
Server.js

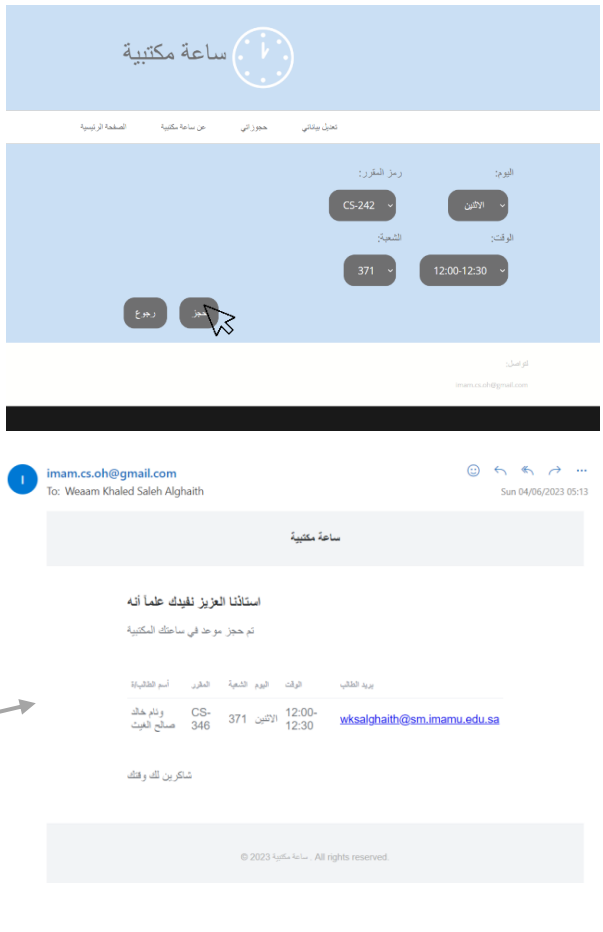
```
app.get('/mybooking', async function(req, res) {
  try{

    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});
  }
});
```

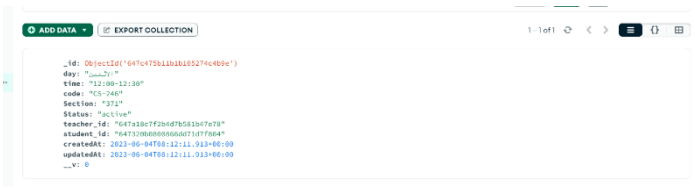
When a user requests the "/mybooking" our website will retrieve all bookings that the user did from a database and then illustrate them on the page. The page has ejs format for the same reason as the index page. As you can see in Figure 24 we store all retrieved data in the reservations list. Additional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







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

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.

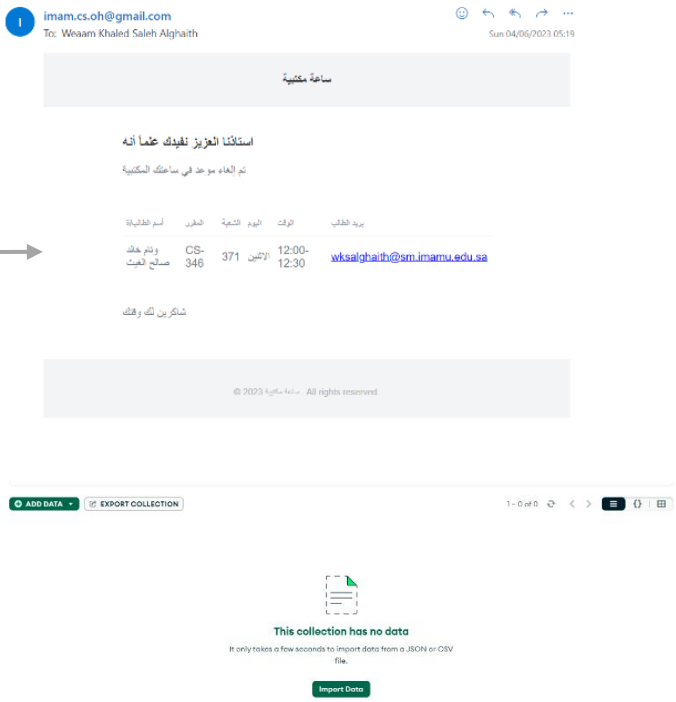


Figure 26. Cancel reservation route.

- 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":
"wksalghaith@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) {
    consol.log(error);
        res.status(400).json({ error });
      }
});
```

Figure 27. Sign in post request.

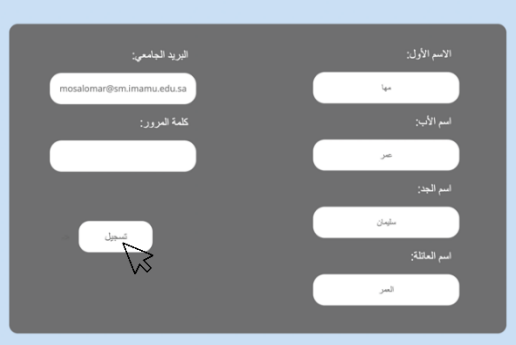
When the user presses the "دخول" the website will take the official email and password from the body request and then search on the database if there is any student who registered with the entered email. If does not exist, it will print an error message "{error: User doesn't exist}". If the user exists, 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.

- 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});
  }

});
```

Figure 28. Sign up post request.



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.