**Booking Management System**

**Abstract**:-

A comprehensive web application called "Booking Management System" was created to simplify user interactions with a unified platform. Admin and User are the two main user roles in the system. Users see a customized dashboard with important data, such as the total number of clients, admins, and users, upon logging in. An overview of bookings, webpages, and user profiles is given via the admin dashboard. Detailed information about bookings is displayed in the admin dashboard, where they are categorized as fulfilled, canceled, or completed. The system also keeps track of the total number of webpages that are created, differentiating between pages that are active and those that are not. A table structure with editing, deleting, and new booking options can be found in the booking section. Booking information, such as the username, booking name, date, and status, can be customized by administrators. Similar features are provided by webpage management, which enables administrators to easily monitor and edit pages. Admins can now create new webpages with attributes like page name, page slug, content, and status thanks to a unique feature that the system offers. Admins can use the provided localhost URL to check the functionality of the webpage. Admins can do tasks akin to bookings and webpages and get insights into registered users through user management. Admins have the ability to add new users and specify their phone number, email address, name, and type of user (user or admin). Individuals can view their booking statistics on a streamlined dashboard with limited functionality. Bookings can be viewed, edited, and deleted from this section, which is a mirror image of the admin interface. Users can update personal information in the user profile section. Constructed with PHP, HTML, CSS, Bootstrap, and the Laravel 10 framework in mind, the Booking Management System offers a centralized platform for effective booking and user management while guaranteeing a user-friendly experience for administrators and users alike.

**Objective**:-

Developing a reliable and user-friendly web application that enables smooth booking and user interaction management is the aim of the "Booking Management System" project. The following goals are the focus of this project:

**Effective Reservations Management**:

Give administrators a centralized platform so they can easily manage reservations, keep track of their status (fulfilled, canceled, or completed), and make new reservations.

**Detailed Dashboard Information:**

Provide useful dashboards with essential data, including the quantity of users, admins, clients, and completed bookings, that are customized for both users and admins.

**Creating and Managing Websites**:

Give administrators the ability to make, edit, and keep an eye on webpages. Include a special feature that lets them create new pages with specific properties like name, slug, content, and status.

**Management of User and Admin Profiles:**

Make it easier to maintain admin and user profiles by enabling the updating of personal data for more customization.

**Friendly User Interface:**

To make navigating simple for administrators and users alike, create a user interface that is both aesthetically pleasing and intuitive.

**Access Control Based on Roles:**

To distinguish between admin and user roles and guarantee that every user has access to the pertinent features and data, use role-based access control..

**Combining Booking Actions:**

For a streamlined and unified user experience, give bookings, webpages, and user management a table-like structure with consistent actions (view, edit, and delete).

**Computerized Stack:**

Use Bootstrap, Laravel 10, HTML, CSS, PHP, and Bootstrap to take advantage of the newest frameworks and technologies for scalable and reliable development.

**Validation of the functionality of a webpage:**

Add a feature that lets administrators visit a specified localhost URL to confirm that newly created webpages are functional.

**Improved Experience for Users:**

Put the user experience first by making sure that it is easy to manage profiles, make new bookings, and browse the application, all of which will contribute to increased user satisfaction.

By fulfilling these goals, the Booking Management System hopes to offer businesses and individuals looking for a complete platform for handling reservations and user communications a strong and effective option.

**Introduction:-**

Effective booking and user engagement management has become critical for businesses and organizations in an era of digital advancements and dynamic interactions. The "Booking Management System" appears as an advanced web application intended to improve and expedite the online booking experience for administrators and users alike.With the goal of offering a centralized and intuitive platform, this all-inclusive system incorporates the newest frameworks and technologies to guarantee scalability, efficiency, and a flawless user experience. This system meets your various needs with accuracy and style, whether you are an administrator looking to manage bookings, webpages, and user profiles with ease, or a user interested in handling personal bookings and profiles. Based on PHP, HTML, CSS, Bootstrap, and powered by the potent Laravel 10 framework, the "Booking Management System" promises a user-friendly interface in addition to strong functionality. The system creates a sense of control and personalization by ensuring that every user, whether an administrator or an end user, has customized access to pertinent features through the use of role-based access control.Come along for a trip where sophistication and efficiency collide as we examine the nuances of booking administration, perceptive dashboards, and the development and validation of webpages. The system encompasses a range of functionalities aimed at improving the user experience in general, from monitoring booking status to dynamically managing user profiles. Take a trip through technology with the "Booking Management System," where user-centric design and innovation meet and online interactions find a beat that fits together. Greetings from a future where handling reservations is an experience rather than a chore.

**Methodology:-**

**Methodology for Authentication:**

**Login Functionality:**

**Objective**: Allow users to log in to the application.

**Steps**:

Display the login form.

Validate user input for email and password.

Use Laravel's Auth facade to attempt user authentication.

Redirect users based on their user type (admin or user).

**Signup Functionality**:

**Objective**: Enable users to register for a new account.

**Steps**:

Display the registration form.

Validate user input for name, phone number, email, and password.

Create a new User model instance and save it to the database.

Redirect users to the login page after successful registration.

**User Profile Management:**

**Objective**: Allow users to view and update their profiles.

**Steps:**

Display the user profile page.

Fetch user information from the database.

Allow users to update their name, email, and phone number.

Save the changes to the database.

**Logout Functionality:**

**Objective**: Enable users to log out of the application.

**Steps**:

Use Laravel's Auth facade to log the user out.

Redirect users to the login page with a success message.

**Methodology for Booking Management :**

**Viewing Bookings:**

**Objective**: Provide admins and users with a list of bookings.

**Steps**:

Fetch bookings from the database, joining with user information.

Display bookings in the admin or user dashboard.

**Adding and Editing Bookings:**

**Objective**: Allow admins and users to add new bookings or edit existing ones.

**Steps**:

Admins can add bookings for any user, while users can only add bookings for themselves.

Display the booking form with appropriate fields.

Validate and save the booking information to the database.

**Deleting Bookings:**

**Objective**: Allow admins and users to delete bookings.

**Steps**:

Display a confirmation view for deleting bookings.

Delete the booking from the database upon confirmation.

Redirect to the appropriate booking list.

**Methodology for User Management :**

**Viewing Users:**

**Objective**: Provide admins with a list of users.

**Steps**:

Fetch users from the database.

Display users in the admin dashboard.

**Adding and Editing Users:**

**Objective**: Allow admins to add new users or edit existing ones.

**Steps**:

Display the user form with appropriate fields.

Validate and save the user information to the database.

**Deleting Users:**

**Objective**: Allow admins to delete users.

**Steps**:

Display a confirmation view for deleting users.

Delete the user from the database upon confirmation.

Redirect to the user list.

**User Profile Management:**

**Objective**: Allow admins and users to view and update their profiles.

**Steps**:

Display the user profile page.

Fetch user information from the database.

Allow users to update their name, email, and phone number.

Save the changes to the database.

**Methodology for WebPage Management :**

**Viewing WebPages:**

**Objective**: Provide admins with a list of webpages.

**Steps**:

Fetch active webpages from the database.

Display webpages in the admin dashboard.

**Adding and Editing WebPages:**

**Objective**: Allow admins to add new webpages or edit existing ones.

**Steps**:

Display the webpage form with appropriate fields.

Validate and save the webpage information to the database.

**Deleting WebPages**:

**Objective**: Allow admins to delete webpages.

**Steps**:

Display a confirmation view for deleting webpages.

Delete the webpage from the database upon confirmation.

Redirect to the webpage list.

**Landing Page and Dynamic Page Viewing:**

**Objective**: Display the landing page and dynamically render content for specific webpages.

**Steps**:

Fetch all webpages for the landing page.

Allow users to view dynamically rendered content for specific webpages.

**Code:-**

**InstallAdmin.php**:

<?php

namespace App\Console\Commands;

use App\Models\User;

use Illuminate\Console\Command;

use Illuminate\Support\Facades\Hash;

class InstallAdmin extends Command

{

    /\*\*

     \* The name and signature of the console command.

     \*

     \* @var string

     \*/

    protected $signature = 'install:admin';

    /\*\*

     \* The console command description.

     \*

     \* @var string

     \*/

    protected $description = 'install System Admin';

    /\*\*

     \* Execute the console command.

     \*/

    public function handle()

    {

        $user = new User([

            'name'=> 'Harsh Satish Kadam',

            'email'=> '20891a1217harshkadam@gmail.com',

            'phone\_no'=>'1234567890',

            'password'=>Hash::make('secret'),

            'user\_type'=>1

        ]);

        if($user->save()){

            $this->info('Account Insert Successfully');

        }

        else{

            $this->info('Account Insert Successfully');

        }

    }

}

**Controllers:**

**AuthController.php:**

<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\DB;

use Illuminate\Http\Request;

use App\Models\User;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Hash;

class AuthController extends Controller

{

    public function login(){

        return view('auth.login');

    }

    public function authenticate(Request $request){

        $request->validate([

            'email' => 'required|email',

            'password'=>'required'

        ]);

        if(Auth::attempt(array('email'=>$request->get('email'),'password'=>$request->get('password')))){

            if(Auth::user()->user\_type == 1){

                return redirect()->route('dashboard.admin');

            }

            else if(Auth::user()->user\_type == 2){

                return redirect()->route('dashboard.user');

            }

        }else{

            return redirect()->route('login')->with('error','Email Address or Password is wrong');

        }

    }

    public function signup(){

        return view('auth.register');

    }

    public function createUser(Request $request){

        $request->validate([

            'name' => 'required|string',

            'phone\_no'=>'required',

            'email' => 'required|email|unique:users',

            'password'=>'required',

            // 'confirmpassword'=>'required'

        ]);

        $user = new User([

            'name'=> $request->get('name'),

            'email'=> $request->get('email'),

            'phone\_no'=>$request->get('phone\_no'),

            'password'=>Hash::make($request->get('password')),

            'user\_type'=>2

        ]);

        $user->save();

        return redirect()->route('login');

    }

    public function logout(){

        Auth::logout();

        return Redirect()->route('login')->withSuccess("Logged out successfully");

    }

}

**BookingController,php:**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\Bookings;

use App\Models\User;

use Illuminate\Support\Facades\Auth;

class BookingController extends Controller

{

    public function index(){

        $query = Bookings::select('bookings.\*','users.name as user\_name');

        $query->LeftJoin('users','bookings.user\_id','=','users.id');

        $data = $query->get();

        return view('AdminDashboard.Bookings.index',['data'=>$data]);

    }

    public function userBookings(){

        $query = Bookings::select('bookings.\*','users.name as user\_name');

        $query->LeftJoin('users','bookings.user\_id','=','users.id');

        $data = $query->get();

        return view('UserDashboard.Bookings.index',['data'=>$data]);

    }

    public function add(){

        $data = User::get();

        return view('AdminDashboard.Bookings.addEdit',['data'=>$data]);

    }

    public function save(Request $request){

        $user = new Bookings([

            'name'=> $request->get('booking\_name'),

            'booking\_datetime'=> $request->get('booking\_on'),

            'status'=>$request->get('booking\_status'),

            'user\_id'=>Auth::user()->user\_type == 1? $request->get('user\_name') : Auth::user()->id

        ]);

        $user->save();

        if(Auth::user()->user\_type == 1){

            $route = 'booking.all';

        }

        else{

            $route='booking.my';

        }

        return redirect()->route($route);

    }

    public function getBookingsById($id){

        $data = User::get();

        $booking =Bookings::find($id);

        return view('AdminDashboard.Bookings.addEdit',['data'=>$data,'booking'=>$booking]);

    }

    public function updateBookingById(Request $request,$id){

        $booking =Bookings::find($id);

        $booking->name = $request->get('booking\_name');

        $booking->booking\_datetime= $request->get('booking\_on');

        $booking->status = $request->get('booking\_status');

        $booking->user\_id=Auth::user()->user\_type == 1? $request->get('user\_name') : Auth::user()->id;

        $booking->save();

        if(Auth::user()->user\_type == 1){

            $route = 'booking.all';

        }

        else{

            $route='booking.my';

        }

        return redirect()->route($route);

    }

    public function viewDelete($id){

        if(Auth::user()->user\_type == 1){

            $route = 'AdminDashBoard.Bookings.delete';

        }

        else{

            $route='UserDashboard.Bookings.delete';

        }

        return view($route);

    }

    public function delete($id){

        $status = Bookings::where('id',$id)->delete();

        if(Auth::user()->user\_type == 1){

            $route = 'booking.all';

        }

        else{

            $route='booking.my';

        }

        return redirect()->route($route);

    }

}

**UserController.php:**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\User;

use App\Models\Bookings;

use App\Models\WebPage;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Hash;

class UserController extends Controller

{

    public function index(){

        $data = User::get();

        return view('AdminDashboard.Users.index',['data'=>$data]);

    }

    public function add(){

        return view('AdminDashboard.Users.addEdit');

    }

    public function save(Request $request){

        $user = new User([

            'name'=> $request->get('name'),

            'email'=> $request->get('email'),

            'phone\_no'=>$request->get('phone\_no'),

            'user\_type'=>$request->get('user\_type')

        ]);

        $user->save();

        return redirect()->route('user');

    }

    public function edit($id){

        $data = User::find($id);

        return view('AdminDashboard.Users.addEdit',['data'=>$data]);

    }

    public function update(Request $request,$id){

        $user = User::find($id);

        $user->name= $request->get('name');

        $user->email= $request->get('email');

        $user->phone\_no=$request->get('phone\_no');

        $user->user\_type=$request->get('user\_type');

        $user->save();

        return redirect()->route('user');

    }

    public function viewDelete($id){

        return view('AdminDashboard.Users.delete');

    }

    public function delete($id){

        User::where('id',$id)->delete();

        return redirect()->route('user');

    }

    public function getProfile(){

        $data = User::find(Auth::user()->id);

        if(Auth::user()->user\_type == 1){

            $route = 'AdminDashboard.Profile.index';

        }

        else{

            $route='UserDashboard.Profile.index';

        }

        return view($route,['data',$data]);

    }

    public function saveProfile(Request $request){

        $user = User::find(Auth::user()->id);

        $user->name= $request->get('name');

        $user->email= $request->get('email');

        $user->phone\_no=$request->get('phone\_no');

        $user->user\_type=$request->get('user\_type');

        $user->save();

        return redirect()->route('user.profile.get');

    }

    public function adminDashBoard(){

        $data['totalUsers'] = 0;

        $data['adminUsers'] = 0;

        $data['clientUsers'] = 0;

        $data['totalBookings'] = 0;

        $data['completedBookings'] = 0;

        $data['cancelledBooking'] = 0;

        $data['fullfilledBooking'] = 0;

        $data['totalWebPages'] = 0;

        $data['activeWebPages']=0;

        $data['totalUsers'] = User::count();

        $data['adminUsers'] = User::where('user\_type',1)->count();

        $data['clientUsers'] = User::where('user\_type',2)->count();

        $data['totalBookings'] = Bookings::count();

        $data['completedBookings'] = Bookings::where('status',1)->count();

        $data['cancelledBooking'] = Bookings::where('status',2)->count();

        $data['fullfilledBooking'] = Bookings::where('status',3)->count();

        $data['totalWebpages'] = WebPage::count();

        $data['activeWebpages']= WebPage::where('status',1)->count();

        return view('AdminDashboard.index',['data'=>$data]);

    }

    public function userDashBoard(){

        $data['totalBookings'] = Bookings::where('user\_id',Auth::user()->id)->count();

        $data['completedBookings'] = Bookings::where('status',1)->where('user\_id',Auth::user()->id)->count();

        return view('UserDashboard.index',['data'=>$data]);

    }

}

**WebPageController.php:**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\WebPage;

use Illuminate\Support\Facades\Auth;

class WebPageController extends Controller

{

    public function index(){

        $data = WebPage::where('status',1)->get();

        return view('AdminDashboard.WebPage.index',['data'=>$data]);

    }

    public function add(){

        return view('AdminDashboard.WebPage.addEdit');

    }

    public function save(Request $request){

        $page = new WebPage([

            'name'=> $request->get('page\_name'),

            'slug'=> $request->get('page\_slug'),

            'html'=>$request->get('page\_content'),

            'status'=>$request->get('page\_status'),

            'created\_by'=>Auth::user()->user\_type

        ]);

        $page->save();

        return redirect()->route('webpage.index');

    }

    public function edit($id){

        $data= WebPage::find($id);

        return view('AdminDashboard.WebPage.addEdit',['data'=>$data]);

    }

    public function update(Request $request,$id){

        $page =WebPage::find($id);

        $page->name= $request->get('page\_name');

        $page->slug= $request->get('page\_slug');

        $page->html=$request->get('page\_content');

        $page->status=$request->get('page\_status');

        $page->updated\_by = Auth::user()->id;

        $page->save();

        return redirect()->route('webpage.index');

    }

    public function viewDelete($id){

        return view('AdminDashboard.WebPage.delete');

    }

    public function delete($id){

        WebPage::where('id',$id)->delete();

        return redirect()->route('webpage.index');

    }

    public function landing(){

        $pages = WebPage::all();

        return view('index', ['pages' => $pages]);

    }

    public function viewPage($page){

        $data = WebPage::where('slug', $page)->first();

        $pages = WebPage::all(); // Fetch all pages (adjust the query as needed)

        return view('dynamic', ['data' => $data, 'pages' => $pages]);

    }

}

**Models:**

**Bookings.php:**

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Bookings extends Model

{

    use HasFactory;

    protected $table="bookings";

    protected $fillable = ['id','name','booking\_datetime','status','user\_id','created\_at','updated\_at'];

}

**User.php:**

<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable

{

    use HasApiTokens, HasFactory, Notifiable;

    /\*\*

     \* The attributes that are mass assignable.

     \*

     \* @var array<int, string>

     \*/

    protected $fillable = [

        'name',

        'email',

        'password',

        'phone\_no',

        'user\_type'

    ];

    /\*\*

     \* The attributes that should be hidden for serialization.

     \*

     \* @var array<int, string>

     \*/

    protected $hidden = [

        'password',

        'remember\_token',

    ];

    /\*\*

     \* The attributes that should be cast.

     \*

     \* @var array<string, string>

     \*/

    protected $casts = [

        'email\_verified\_at' => 'datetime',

        'password' => 'hashed',

    ];

}

**WebPage.php:**

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class WebPage extends Model

{

    use HasFactory;

    protected $table = "webpage";

    protected $fillable = ['name','slug','html','status','created\_by','updated\_by','created\_at','updated\_at'];

}

**Views:**

**Admin DashBoard:**

**Index.blade.php(I changed it some):**

@extends('AdminDashboard.Layout.adminBaseView')

<style>

    .icon {

  float: right;

  font-size:500%;

  position: absolute;

  top:0rem;

  right:-0.3rem;

  opacity: .16;

}

#user-container

{

  width: 1200px;

  display: flex;

}

#web-container

{

  width: 1200px;

  display: flex;

}

#booking-container

{

  width: 1200px;

  display: flex;

}

.grey-dark

{

  background: #495057;

  color: #efefef;

}

.red-gradient {

  background: linear-gradient(180deg, rgba(207,82,82,1) 0%, rgba(121,9,9,1) 80%);

  color: #fff;

}

.red {

  background: #a83b3b;

  color: #fff;

}

.purple

{

  background: #886ab5;

  color: #fff;

}

.orange {

  background: #ffc241;

  color: #fff;

}

.kpi-card

{

  overflow: hidden;

  position: relative;

  box-shadow: 1px 1px 3px rgba(0,0,0,0.75);;

  display: inline-block;

  float: left;

  padding: 1em;

  border-radius: 0.3em;

  font-family: sans-serif;

  width: 240px;

  min-width: 180px;

  margin-left: 0.5em;

  margin-top: 0.5em;

}

.card-value {

  display: block;

  font-size: 200%;

  font-weight: bolder;

}

.card-text {

  display:block;

  font-size: 70%;

  padding-left: 0.2em;

}

</style>

@section('dashContent')

    <div class="container mt-2">

        <h6>User KPI</h6><hr>

        <div id="user-container">

            <div class="kpi-card orange">

                <span class="card-value">{{ isset($data['totalUsers']) ? $data['totalUsers'] : 0 }}</span>

                <span class="card-text">Total Users</span>

                <i class="bi bi-people-fill icon"></i>

            </div>

            <div class="kpi-card orange">

                <span class="card-value">{{ isset($data['adminUsers']) ? $data['adminUsers'] : 0 }}</span>

                <span class="card-text">Admin Users</span>

                <i class="bi bi-person-lock icon"></i>

            </div>

            <div class="kpi-card orange">

                <span class="card-value">{{ isset($data['clientUsers']) ? $data['clientUsers'] : 0 }} </span>

                <span class="card-text">Client Users</span>

                <i class="bi bi-person icon"></i>

            </div>

        </div>

    </div>

    <div class="container  mt-2">

        <h6>Booking KPI</h6><hr>

        <div id="web-container">

            <div class="kpi-card purple">

                <span class="card-value">{{ isset($data['totalBookings']) ? $data['totalBookings'] : 0 }}</span>

                <span class="card-text">Total Bookings</span>

                <i class="bi bi-journal-plus icon"></i>

            </div>

            <div class="kpi-card grey-dark">

                <span class="card-value">{{ isset($data['completedBookings']) ? $data['completedBookings'] : 0 }}</span>

                <span class="card-text">Completed Bookings</span>

                <i class="bi bi-journal-check icon"></i>

            </div>

            <div class="kpi-card grey-dark">

                <span class="card-value">{{ isset($data['cancelledBooking']) ? $data['cancelledBooking'] : 0 }}</span>

                <span class="card-text">Booking Cancelled</span>

                <i class="bi bi-journal-check icon"></i>

            </div>

            <div class="kpi-card grey-dark">

                <span class="card-value">{{ isset($data['fullfilledBooking']) ? $data['fullfilledBooking'] : 0 }}</span>

                <span class="card-text">Booking Fullfilled</span>

                <i class="bi bi-journal-check icon"></i>

            </div>

        </div>

    </div>

    <div class="container  mt-2">

        <h6>Webpage KPI</h6><hr>

        <div id="booking-container">

            <div class="kpi-card red">

                <span class="card-value">{{ isset($data['totalWebpages']) ? $data['totalWebpages'] : 0 }}</span>

                <span class="card-text">Total Web Pages</span>

                <i class="bi bi-globe icon"></i>

            </div>

            <div class="kpi-card red">

                <span class="card-value">{{ isset($data['activeWebpages']) ? $data['activeWebpages'] : 0 }}</span>

                <span class="card-text">Active Web Pages</span>

                <i class="bi bi-clipboard2-check icon"></i>

            </div>

        </div>

    </div>

@endsection

**Routes:**

**Web.php**

<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\AuthController;

use App\Http\Controllers\BookingController;

use App\Http\Controllers\UserController;

use App\Http\Controllers\WebPageController;

/\*

|--------------------------------------------------------------------------

| Web Routes

|--------------------------------------------------------------------------

|

| Here is where you can register web routes for your application. These

| routes are loaded by the RouteServiceProvider and all of them will

| be assigned to the "web" middleware group. Make something great!

|

\*/

// Route::get('/', function () {

//     return view('welcome');

// });

//Web Site Routes

Route::get('/',[WebPageController::class,'landing'])->name('webpage.view');

Route::get('page/{name}',[WebPageController::class,'viewPage'])->name('webpage.dynamic');

//Authorization Routes

Route::get('login',[AuthController::class,'login'])->name('login');

Route::post('login',[AuthController::class,'authenticate'])->name('login.authenticate');

Route::get('signup',[AuthController::class,'signup'])->name('signup');

Route::post('signup',[AuthController::class,'createUser'])->name('signup.create');

Route::get('logout',[AuthController::class,'logout'])->name('logout');

//Authenticate Routes for admin and users

Route::middleware(['auth'])->group(function(){

    //DashBoard Route for Admin

    Route::get('dashboard/admin',[UserController::class,'adminDashBoard'])->name('dashboard.admin');

    //DashBoard Route for Users

    Route::get('dashboard/user',[UserController::class,'userDashBoard'])->name('dashboard.user');

    //Booking Related Routes

    Route::get('booking/all',[BookingController::class,'index'])->name('booking.all');

    Route::get('booking/my',[BookingController::class,'userBookings'])->name('booking.my');

    Route::get('booking/add',[BookingController::class,'add'])->name('booking.add');

    Route::post('booking/save',[BookingController::class,'save'])->name('booking.save');

    Route::get('booking/{id}',[BookingController::class,'getBookingsById'])->name('booking.edit');

    Route::post('booking/{id}',[BookingController::class,'updateBookingById'])->name('booking.update');

    Route::get('booking/delete/{id}',[BookingController::class,'viewDelete'])->name('booking.view.delete');

    Route::post('booking/delete/{id}',[BookingController::class,'delete'])->name('booking.delete');

    //WebPage Related Routes

    Route::get('webpage',[WebPageController::class,'index'])->name('webpage.index');

    Route::get('webpage/add',[WebPageController::class,'add'])->name('webpage.add');

    Route::post('webpage/save',[WebPageController::class,'save'])->name('webpage.save');

    Route::get('webpage/{id}',[WebPageController::class,'edit'])->name('webpage.edit');

    Route::post('webpage/{id}',[WebPageController::class,'update'])->name('webpage.update');

    Route::get('webpage/delete/{id}',[WebPageController::class,'viewDelete'])->name('webpage.view.delete');

    Route::post('webpage/delete/{id}',[WebPageController::class,'delete'])->name('webpage.delete');

    //User Related Routes

    Route::get('user',[UserController::class,'index'])->name('user');

    Route::get('user/add',[UserController::class,'add'])->name('user.add');

    Route::post('user/save',[UserController::class,'save'])->name('user.save');

    Route::get('user/{id}',[UserController::class,'edit'])->name('user.edit');

    Route::post('user/{id}',[UserController::class,'update'])->name('user.update');

    Route::get('user/delete/{id}',[UserController::class,'viewDelete'])->name('user.view.delete');

    Route::post('user/delete/{id}',[UserController::class,'delete'])->name('user.delete');

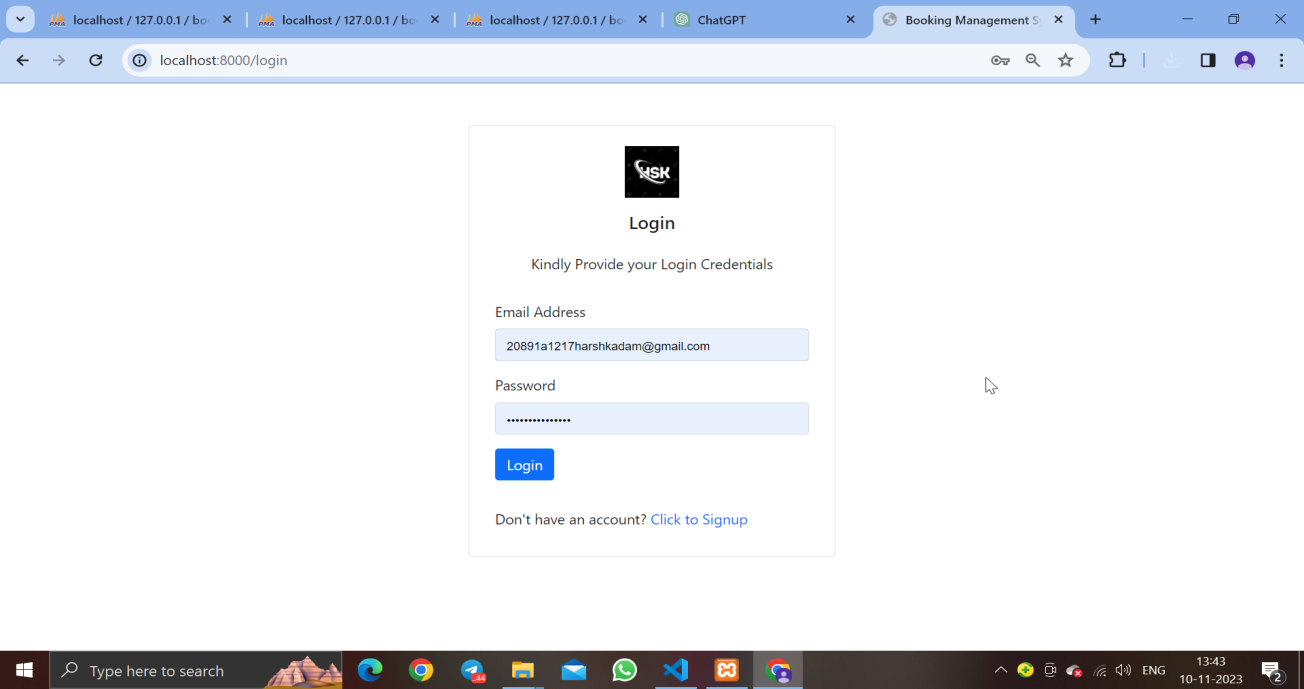
    //user profile related routes

    Route::get('profile',[UserController::class,'getProfile'])->name('user.profile.get');

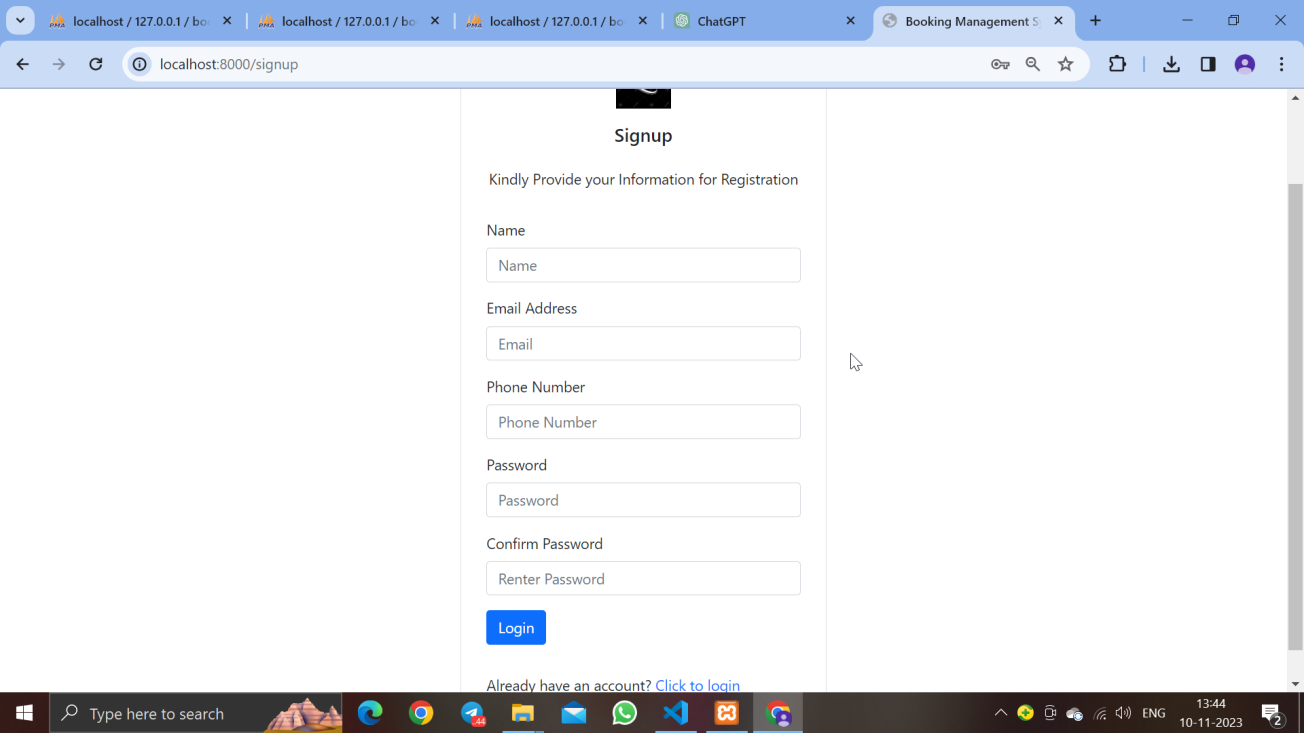
    Route::post('profile',[UserController::class,'saveProfile'])->name('user.profile.save');

});

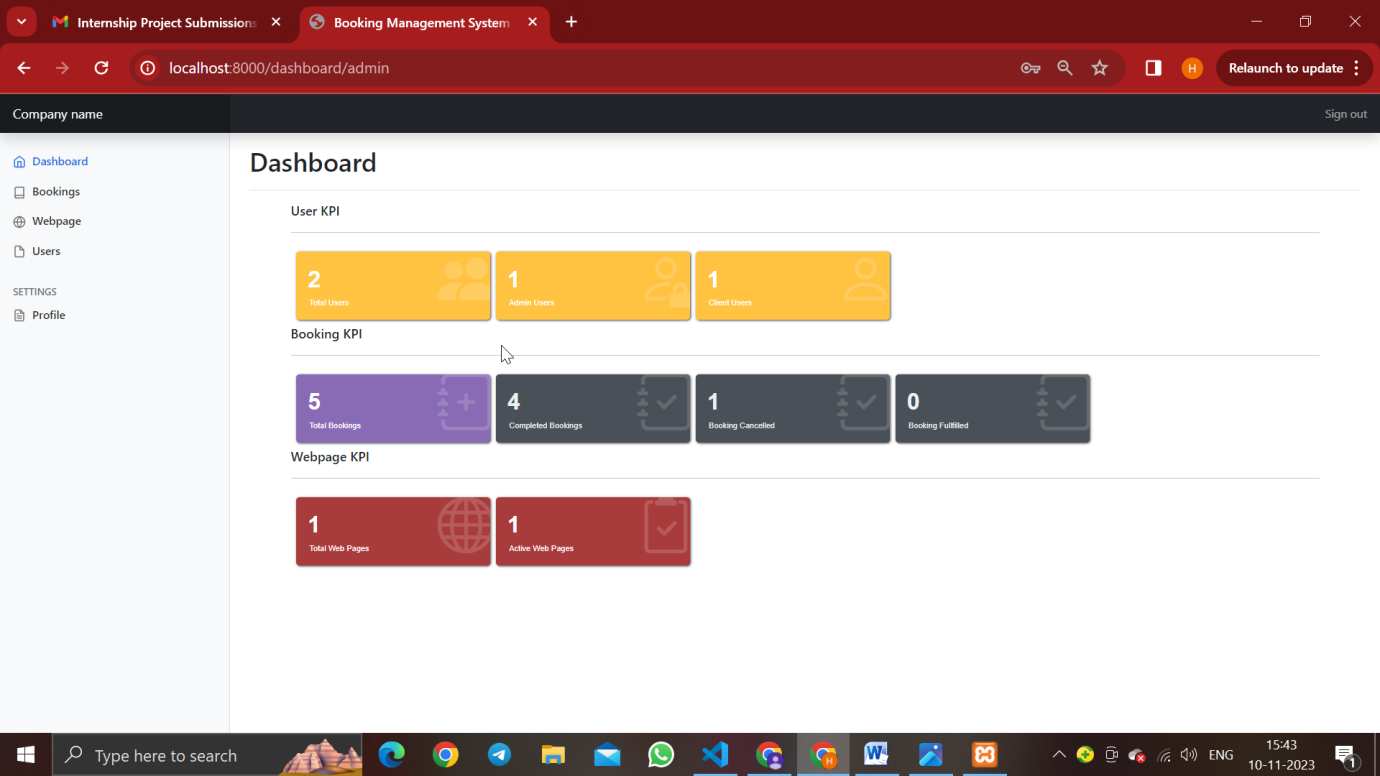
**Output of the Following code are:**

****

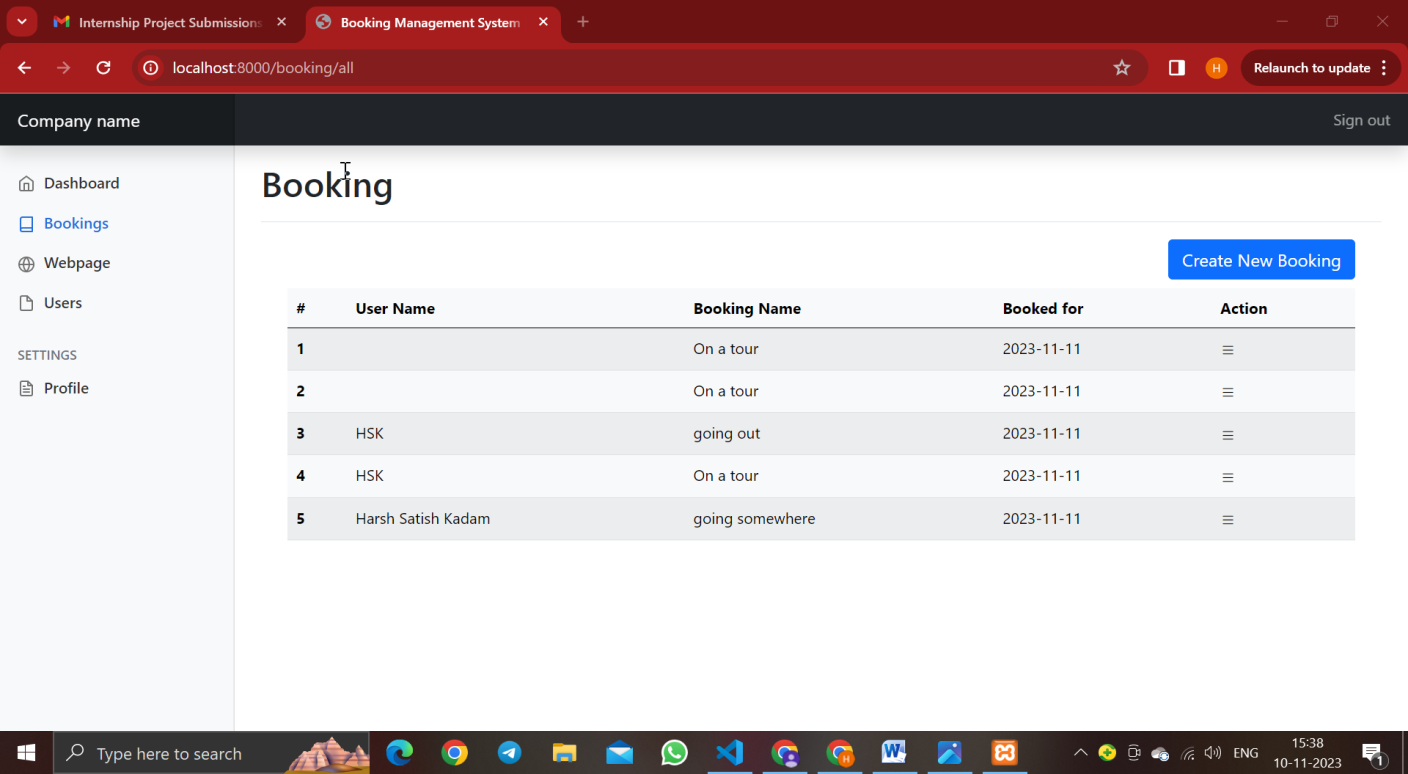
**Login page**

****

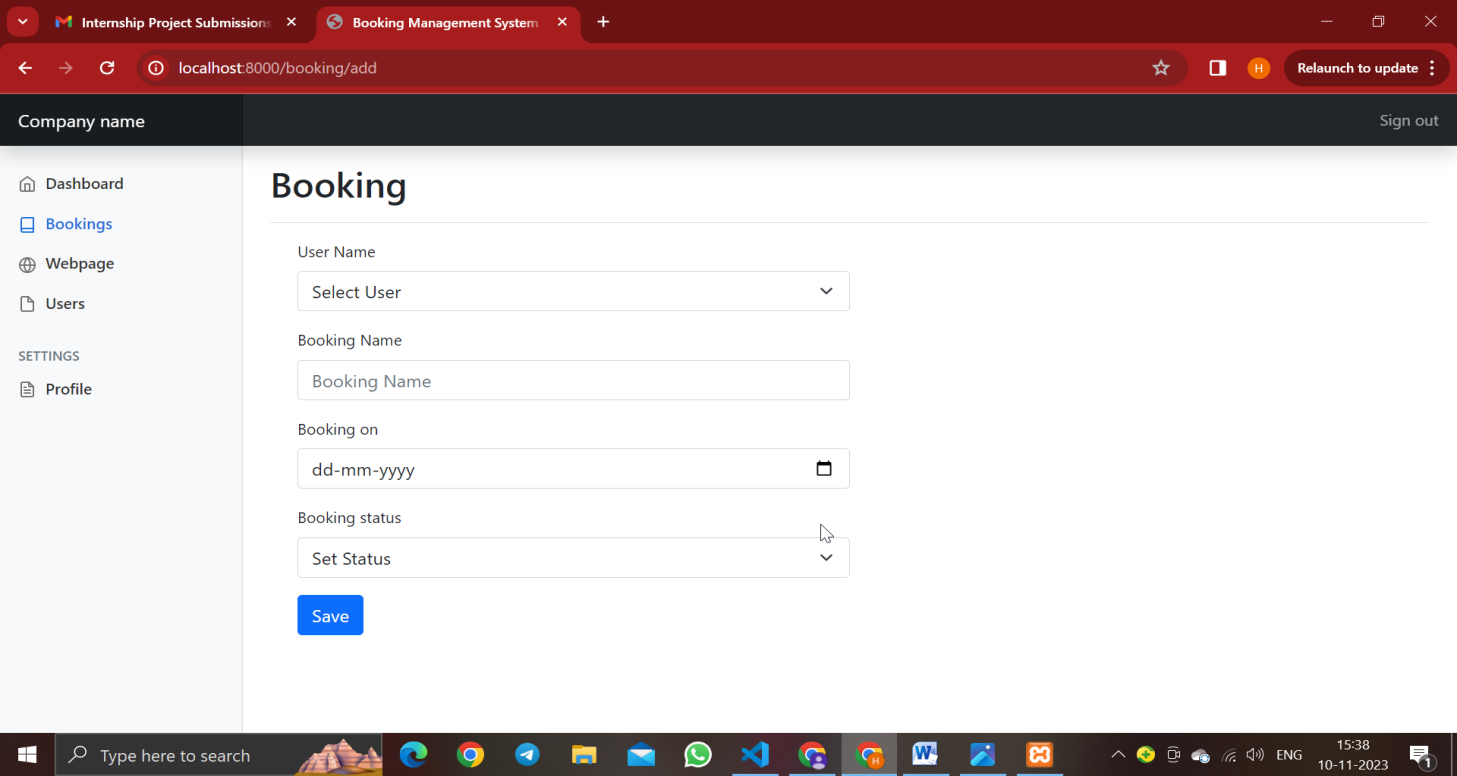
Registration form

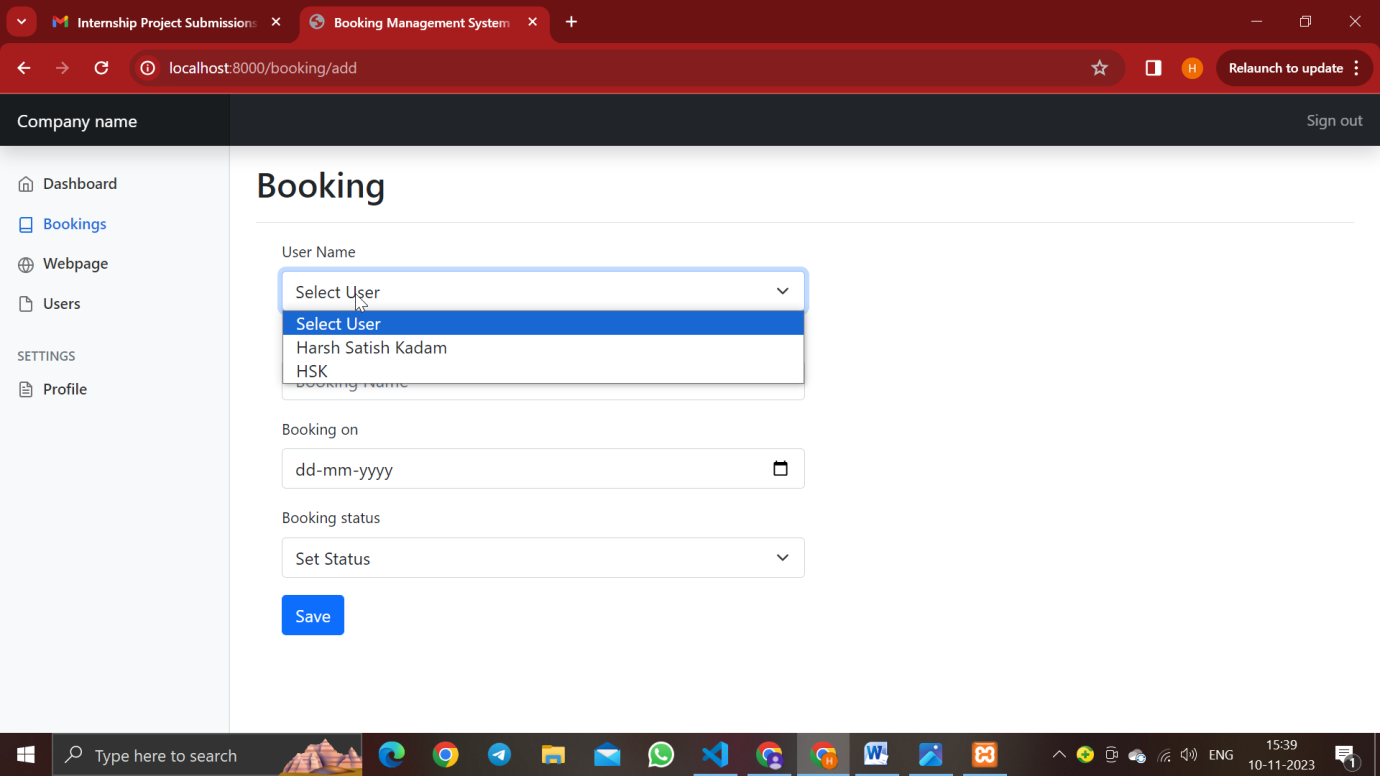


Admin DashBoard

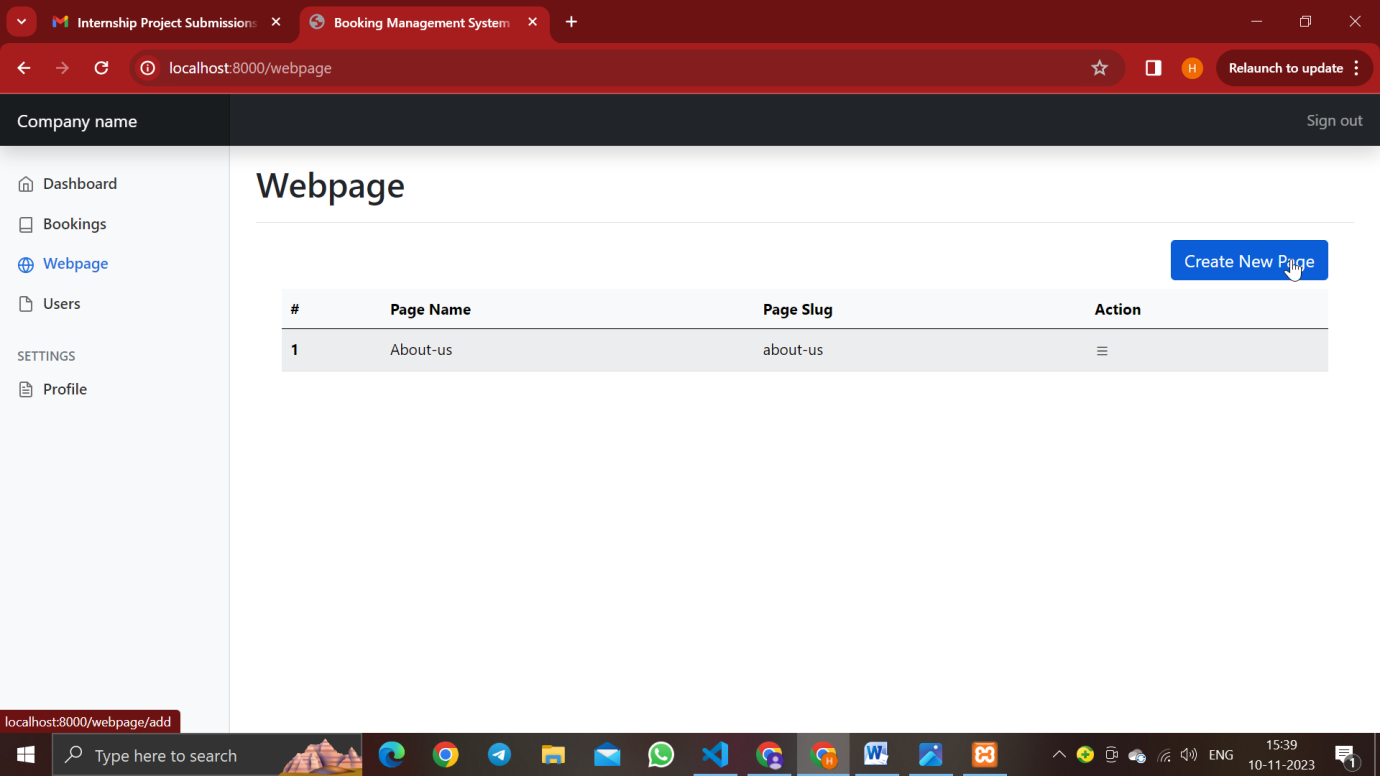


Bookings page

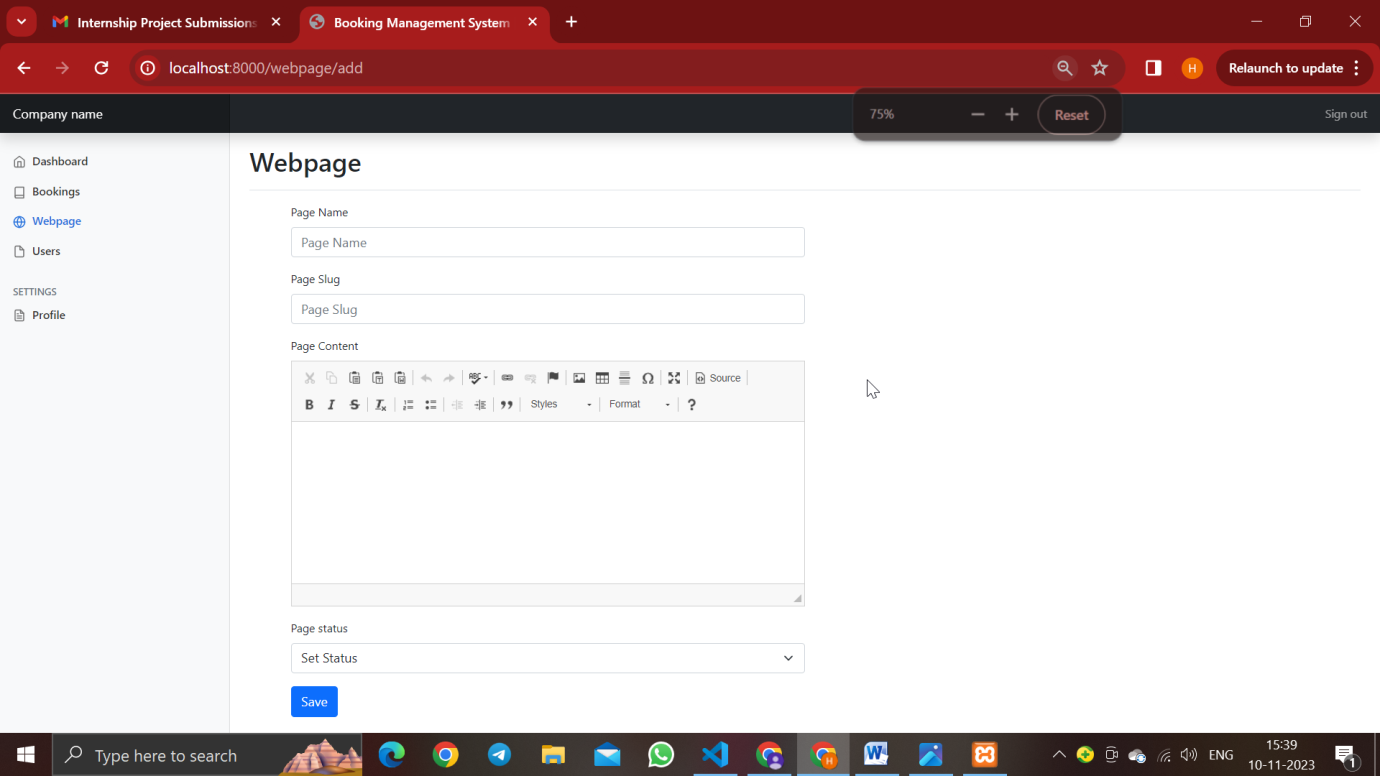




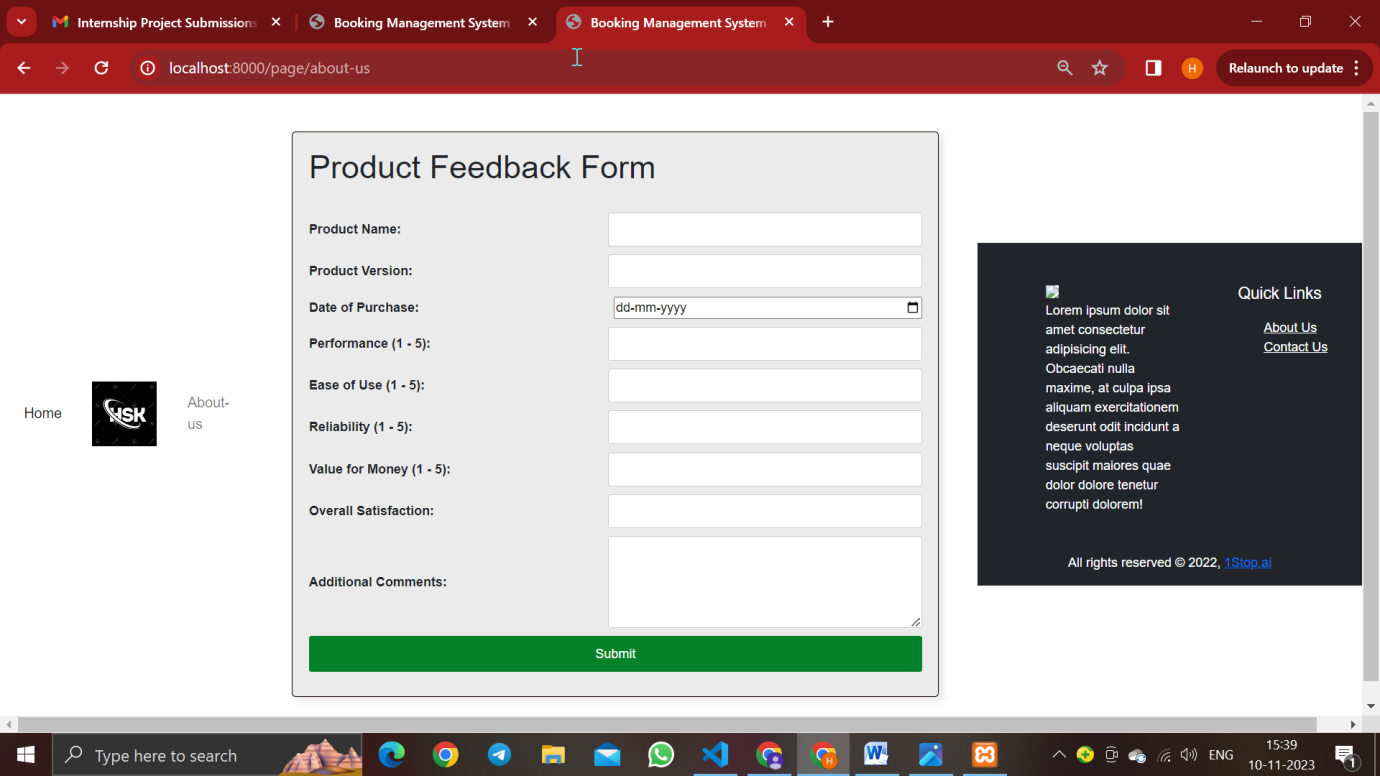
Booking Portal



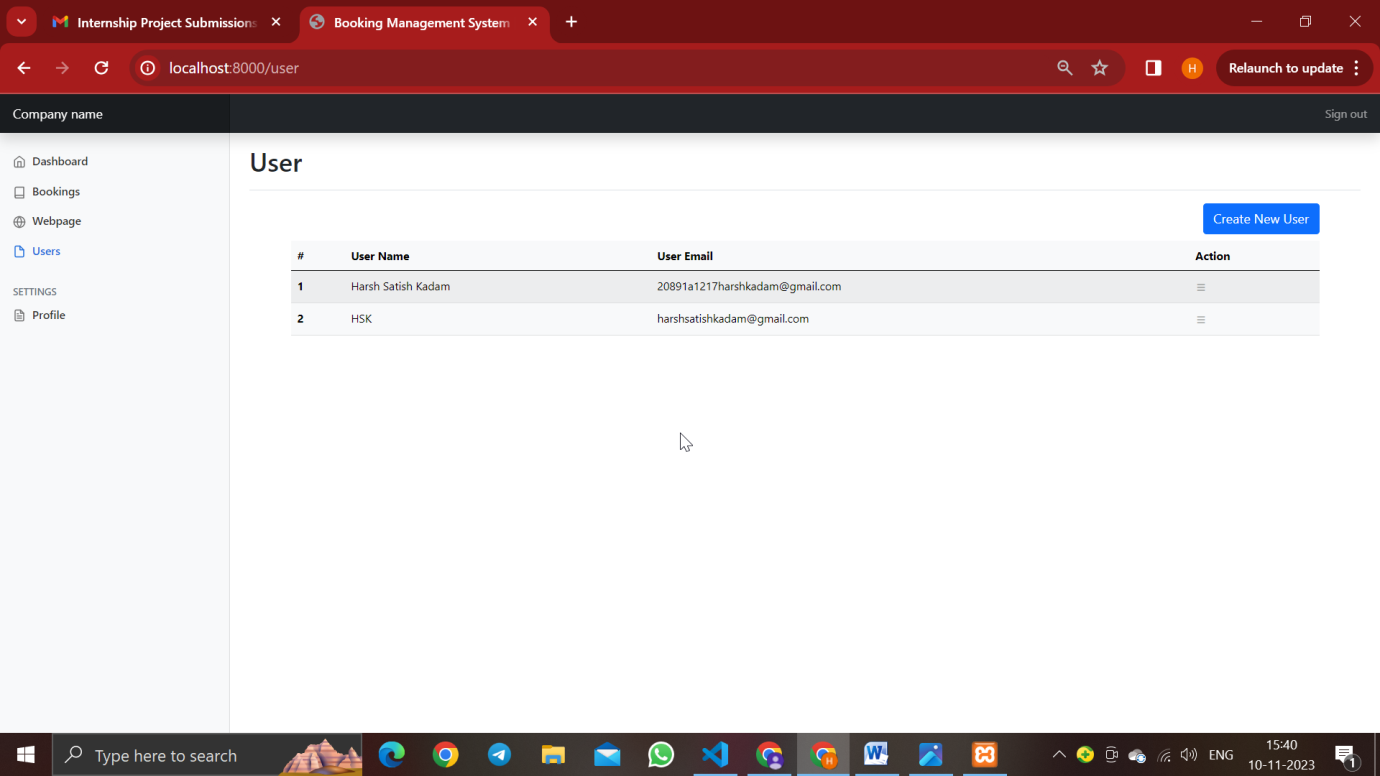
WebPage Portal



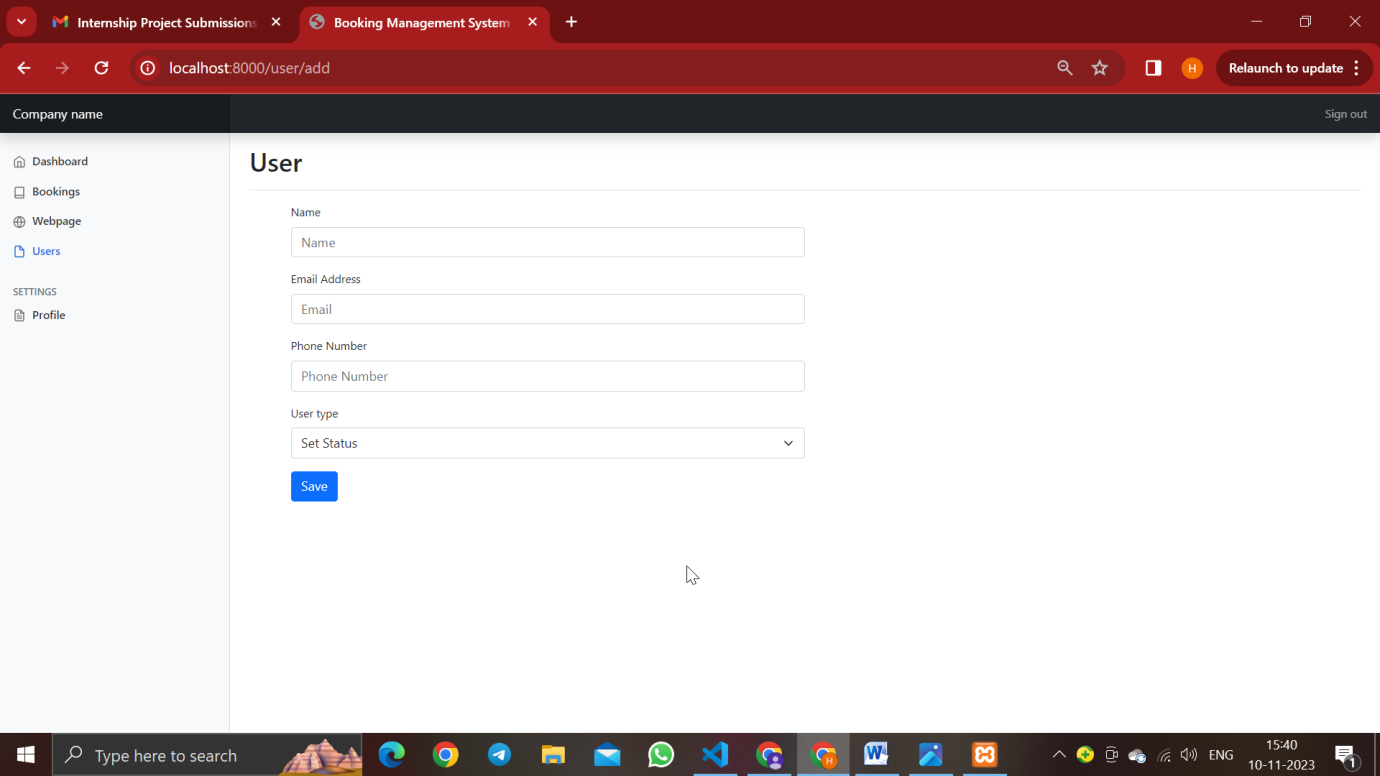
WebPage Portal



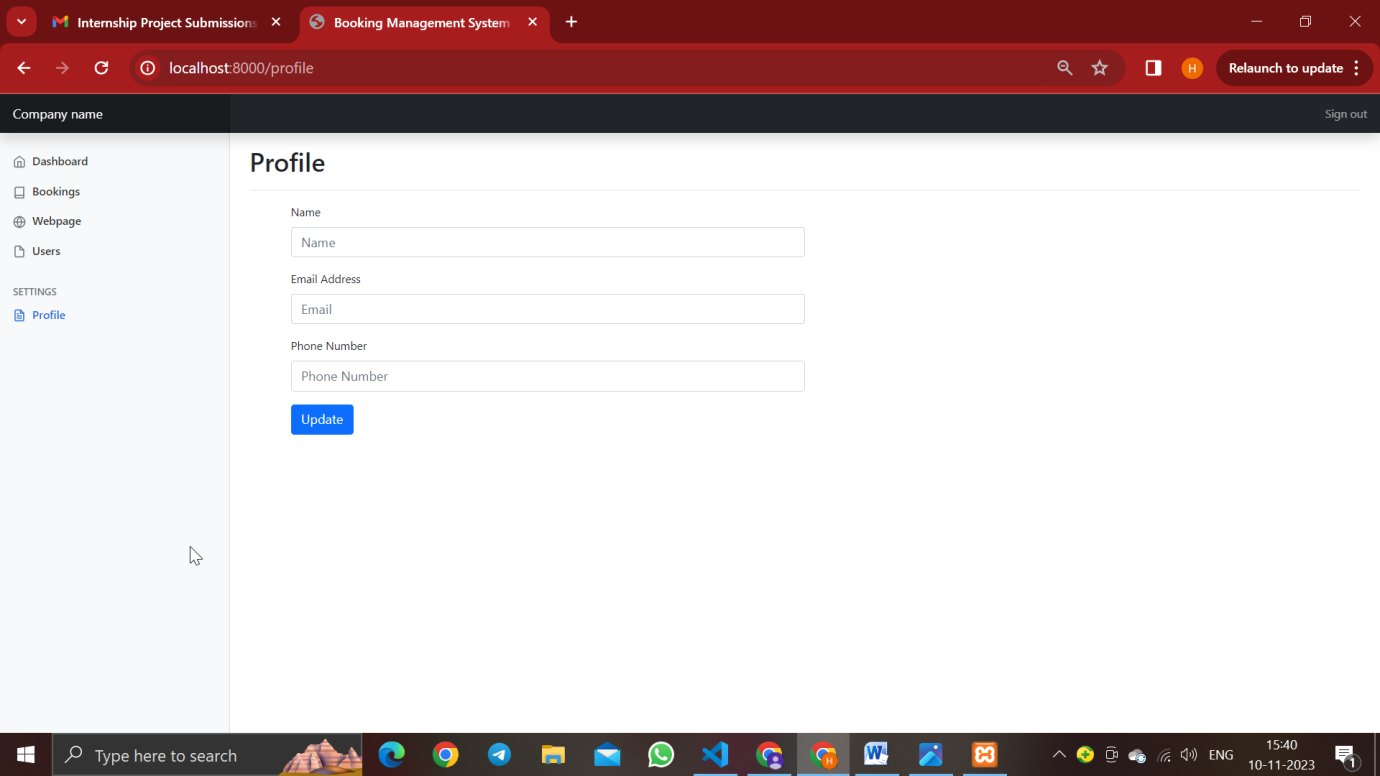
The webpage when run.



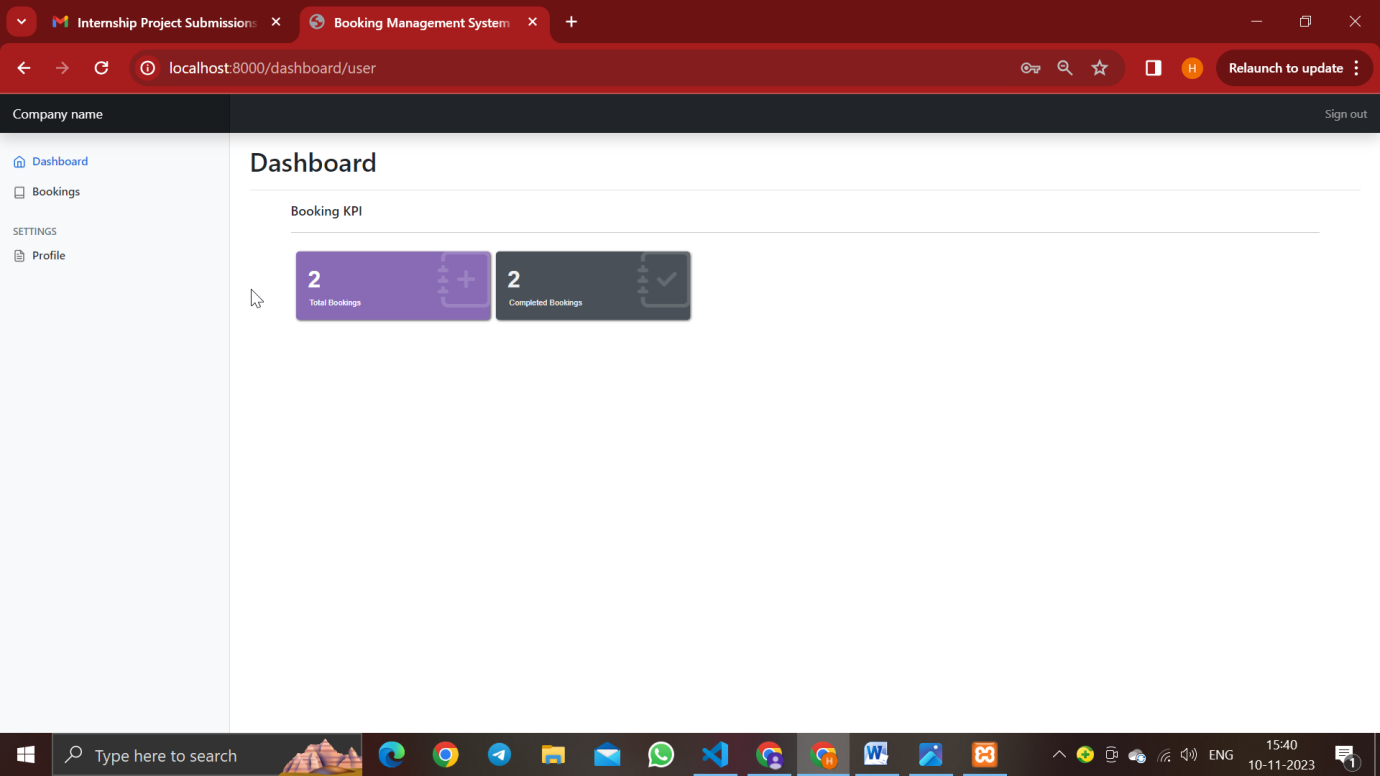
User Portal



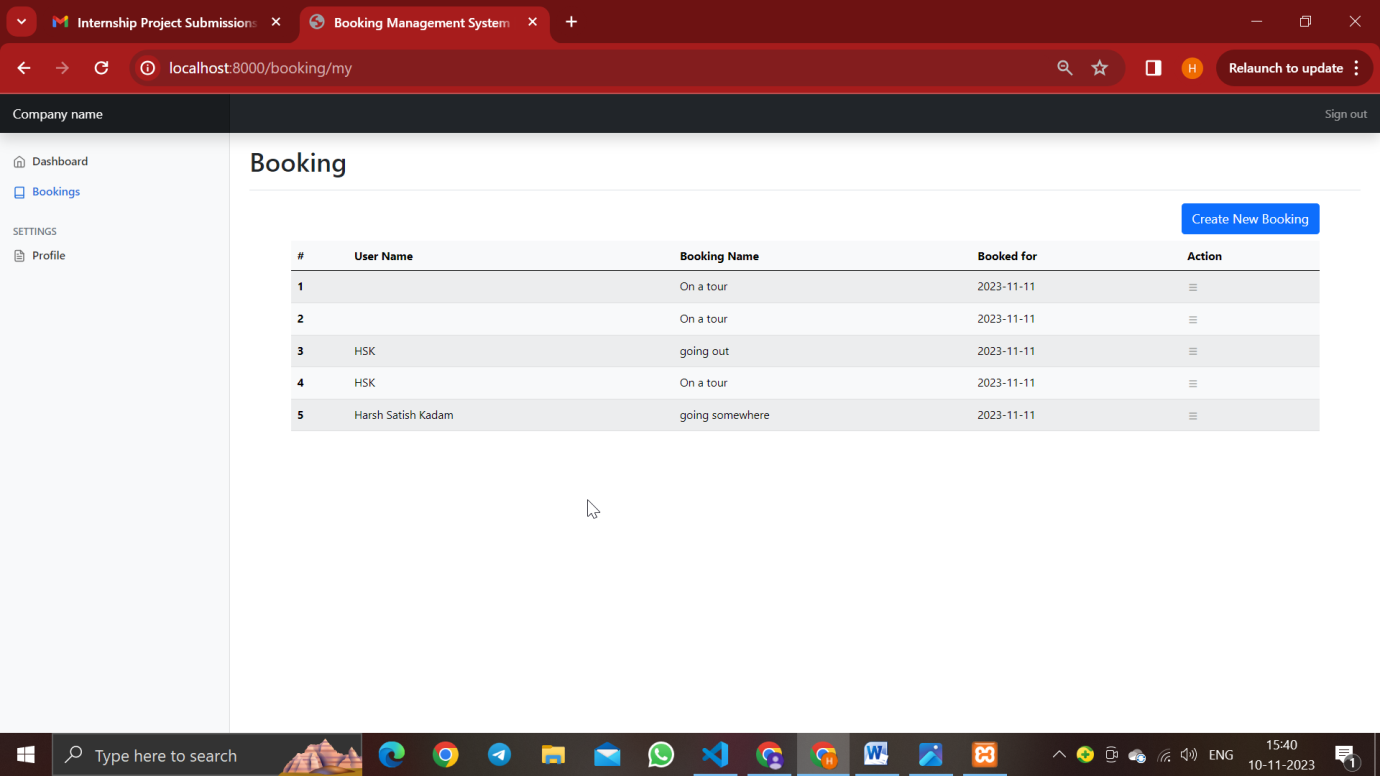
User Portal



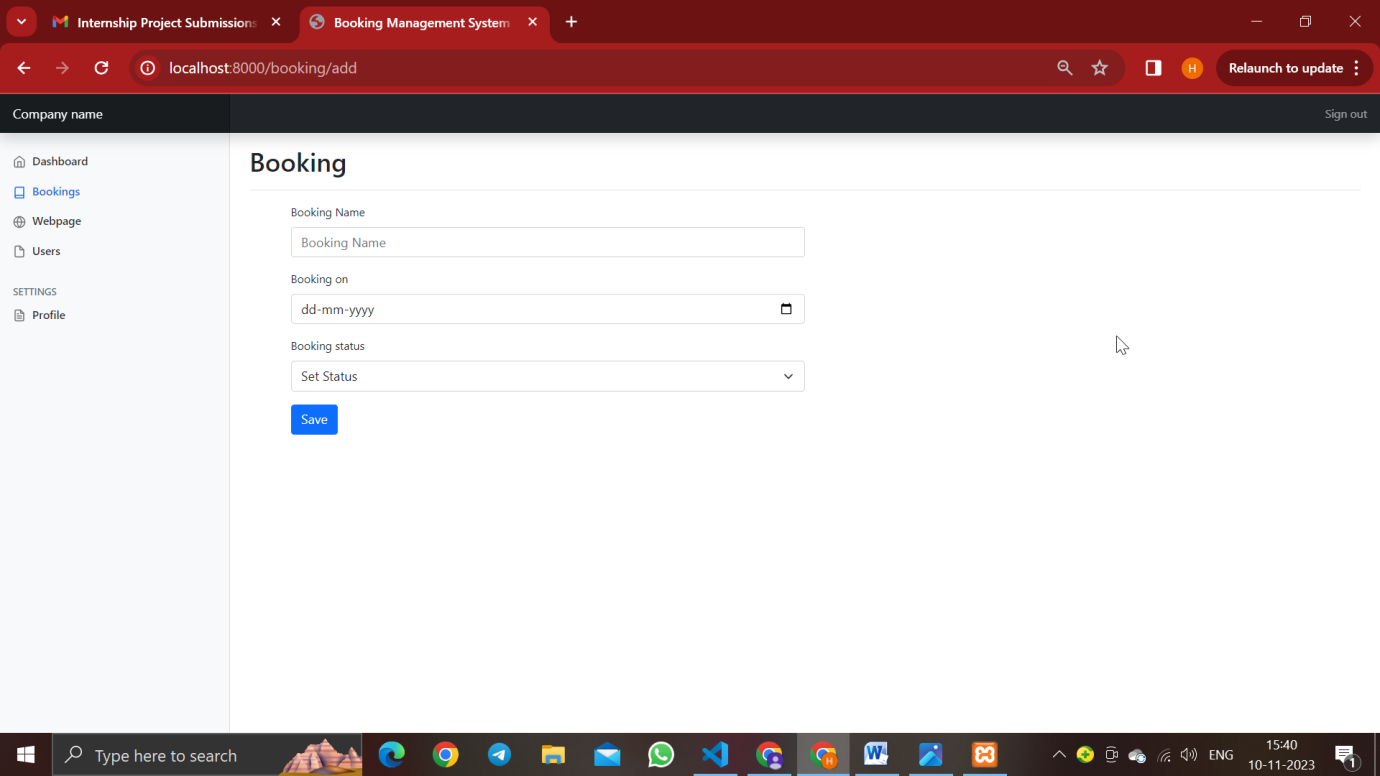
Profile Portal(Admin side).



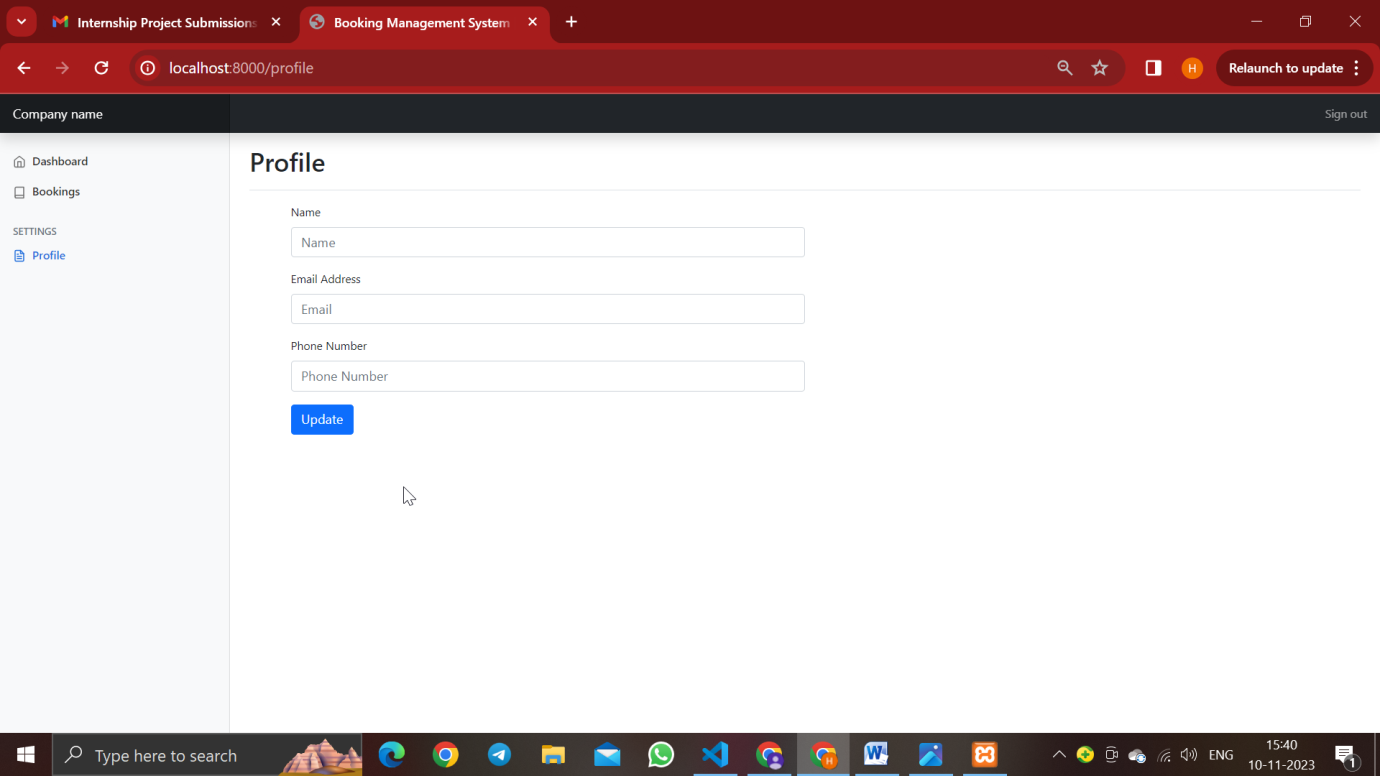
Client Side DashBoard.



Client Side Booking System.



Client Side Booking portal(Once saved will automatically remove the webpage and users from the hamburger menu).



The Client Side Profile Portal.

**Conclusion**:-

Throughout the "Booking Management System" development process, careful attention to detail, deliberate design, and calculated implementation have come together to produce a reliable and user-focused web application. The system, which is controlled by Laravel 10, is an example of contemporary web development techniques. It makes use of Bootstrap, HTML, CSS, and PHP to provide a feature-rich and user-friendly interface.The AuthController leads the authentication process, which guarantees safe user access and allows admin and user dashboard navigation to be smooth. This methodology offers a safe and effective basis for user interaction by concentrating on user-centric actions like login, signup, user profile management, and logout. The precise management of bookings is guided by the BookingController methodology. With the ability to view, edit, and delete bookings with ease, the system meets the various needs of administrators and users alike. Bookings can be added, edited, and deleted with ease thanks to the BookingController, which promotes an efficient workflow.The methodology of UserController offers a thorough method for managing users. The system makes sure administrators have the resources they need to effectively manage user data, from user listing to profile management. The UserController's dual function, meeting the needs of administrators and users alike, demonstrates the system's adaptability. WebPageController delves into webpage management with its methodology. Administrators can easily manage the addition, modification, and removal of webpages, guaranteeing users dynamic and interesting content. The user experience is improved overall by the landing page and dynamic page viewing features, which offer a higher level of sophistication.Within this methodology's grandeur is a project that not only achieves its goals but also establishes a standard for effective, safe, and user-friendly web applications. By carefully coordinating these controllers and having a deep comprehension of user journeys, the "Booking Management System" stands out as a superior booking and content management solution. The unwavering commitment to industry best practices, user satisfaction, and continuous improvement is evident throughout the development process. Not just a project, the "Booking Management System" is a demonstration of how well technology and user-centric design can work together. It is expected to have a significant influence on the market for online booking systems.

Git Hub Link:- <https://github.com/Harshkadam1217/Projects-1/tree/main/Booking%20Management%20System>