

**CIS 5669: Communication Network Management**

**Spring 2024**

A Project on

**Kansas Camp Adventure**

Sayali Prakash Kadam (#700751777)

Pavankumar Reddy Kunduru (#700760406)

Prashanthi Repakula (#700754388)

Submission Date: **04/20/2024**

**Abstract:**

The camping project is a website designed to simplify the planning and booking process for camping trips. In today's dynamic landscape, where outdoor activities and recreational experiences are in high demand, camp owners require efficient tools to streamline operations and enhance customer satisfaction. The KC Adventure Camp Scheduling System addresses these needs by providing a comprehensive solution for managing campgrounds, scheduling bookings, and facilitating communication between camp owners and attendees.

At the core of the Camp Scheduling System is a powerful backend server built on Python and Flask, which handles data processing, business logic, and communication with external services. The frontend application, developed using HTML, CSS, and JavaScript, offers an intuitive and responsive interface for camp owners and attendees to interact with the system. Leveraging modern web technologies, the frontend application provides seamless navigation, real-time updates, and support for various devices and screen sizes.

Key features of the system include:

1. **Campground Management:** Camp owners can easily list their campgrounds, including detailed descriptions, availabilities, and pricing information. They can update availability calendars, set booking restrictions, and manage campground profiles to attract more visitors.
2. **Booking Scheduling:** Attendees can search for campgrounds based on location, availability, and their budget. The system provides a streamlined booking process, allowing users to reserve campsites and select preferred dates according to their budget.
3. **User Authentication and Authorization:** Secure user authentication ensures that only authorized individuals can access the system. Camp owners and attendees have personalized accounts with unique login credentials, allowing them to manage bookings, view past reservations, and update profile information.
4. **Administrative Capabilities:** Camp owners have access to administrative tools for managing user accounts, reviewing booking requests, and generating reports. They have access to add properties according to their management and manage pricings.

In summary, the Camp Scheduling System offers a comprehensive solution for camp owners and attendees to efficiently manage campgrounds, schedule bookings, and enhance the overall camping experience. Our project touches all the components of functional application, including the VPC, internet gateway, route table, subnets, security group, key pair, EC2 instance, S3 bucket, and RDS instance.

**Python Boto3 Automation:**

1. We have created a new user in IAM and have added policies such as AmazonRDSFullAccess, AmazonEC2FullAccess, AmazonS3FullAccess, AmazonVPCFullAccess, and IAMFullAccess.

A screenshot of a computer

Description automatically generated

1. We've generated an access key within the security credentials section of the IAM user interface. These credentials will enable us to utilize Boto3 services within our local Python code.

A screenshot of a computer

Description automatically generated

1. We have automated the process of creating VPC by writing code to create VPC as provided in zipped file (kc\_adventure\_script.py). We have used create\_vpc function, setting its CIDR block to "10.0.0.0/16".

A screenshot of a computer

Description automatically generated

1. To create internet gateway for VPC we have used create\_internet\_gateway function and to attach it to VPC we have used attach\_internet\_gateway function.

A screenshot of a computer

Description automatically generated

1. For creating routing table for VPC we have used create\_route\_table function. This create\_route function is used point at internet gateway by adding default route allowing instances in VPC to establish communication with internet.

A screenshot of a computer

Description automatically generated

1. The create\_subnet function is creating 2 subnets, with CIDR blocks designated as "10.0.1.0/24" and "10.0.2.0/24" respectively. Using associate\_route\_table function, route tables are associated with these subnets. A screenshot of a computer

   Description automatically generated
2. Using authorize\_security\_group\_ingress we have created security group allowing inbound traffic for SSH, HTTP, and HTTPS protocols. A screenshot of a computer

   Description automatically generated
3. To authenticate access to EC2 instance we need key pair. For this a key\_pair is assigned a name for key pair. Then through code we try to fetch the key pair by using describe\_key\_pairs() method. If key exist we use same key but if key doesn’t match then code proceeds with new create\_key\_pair() method and stores it in a local file.

A screenshot of a computer

Description automatically generated

1. The instance is launched with specified AMI ID and it is associated with network interface within designated subnet and security group. The server\_name variable is assigned a name for the EC2 instance. The code then verifies if an instance with the same name already exists by utilizing the describe\_instances() method. If an instance is found, it retrieves the instance ID. Otherwise, it proceeds to create a new instance using the run\_instances() method.

A screenshot of a computer

Description automatically generated

1. This code creates a S3 bucket and sets bucket policies to make object public. It also enable static website hosting for bucket by calling put\_bucket\_website(). The code first checks if the S3 bucket already exist and if it doesn’t exist then the code creates S3 bucket using the specified bucket name. Then the code prints URL for static website hosted in S3 bucket.

A screenshot of a computer

Description automatically generated

1. Using create\_db\_subnet\_group method within the RDS client a new subnet group is created.

A screenshot of a computer

Description automatically generated

1. Now using create\_db\_instance method we create new RDS instance. This method encompasses various parameters configuring the instance, including the database engine (Engine), instance class (DBInstanceClass), and database name (DBName). The VpcSecurityGroupIds parameter designates the security group to employ, while the DBSubnetGroupName parameter specifies the subnet group. The PubliclyAccessible parameter is set to True, enabling access from the Internet. A screenshot of a computer

   Description automatically generated
2. The final phase of the kc adventure involves connecting to the EC2 instance, setting up nginx, and installing the required packages for the backend server. The EC2 instance's IP address is then utilized within the frontend code to establish connection with the server. This frontend code is subsequently deployed within the S3 bucket previously created. A screenshot of a computer

   Description automatically generated

**Technologies used.**

* JavaScript, CSS, HTML
* Python Boto3
* AWS Services IAM, VPC, EC2, S3 and RDS.

**Functionalities**

* Camp owners can log in and manage their Camping location availability, Camping activities, pricing, and other details.
* Customers can search for available camping location and schedule bookings based on their preferences.
* Customers can view location details, including pricing, availability, and activities.
* Camp owners(admin) can manage their bookings and approve or reject requests from customers.

**KC\_Adventure App**

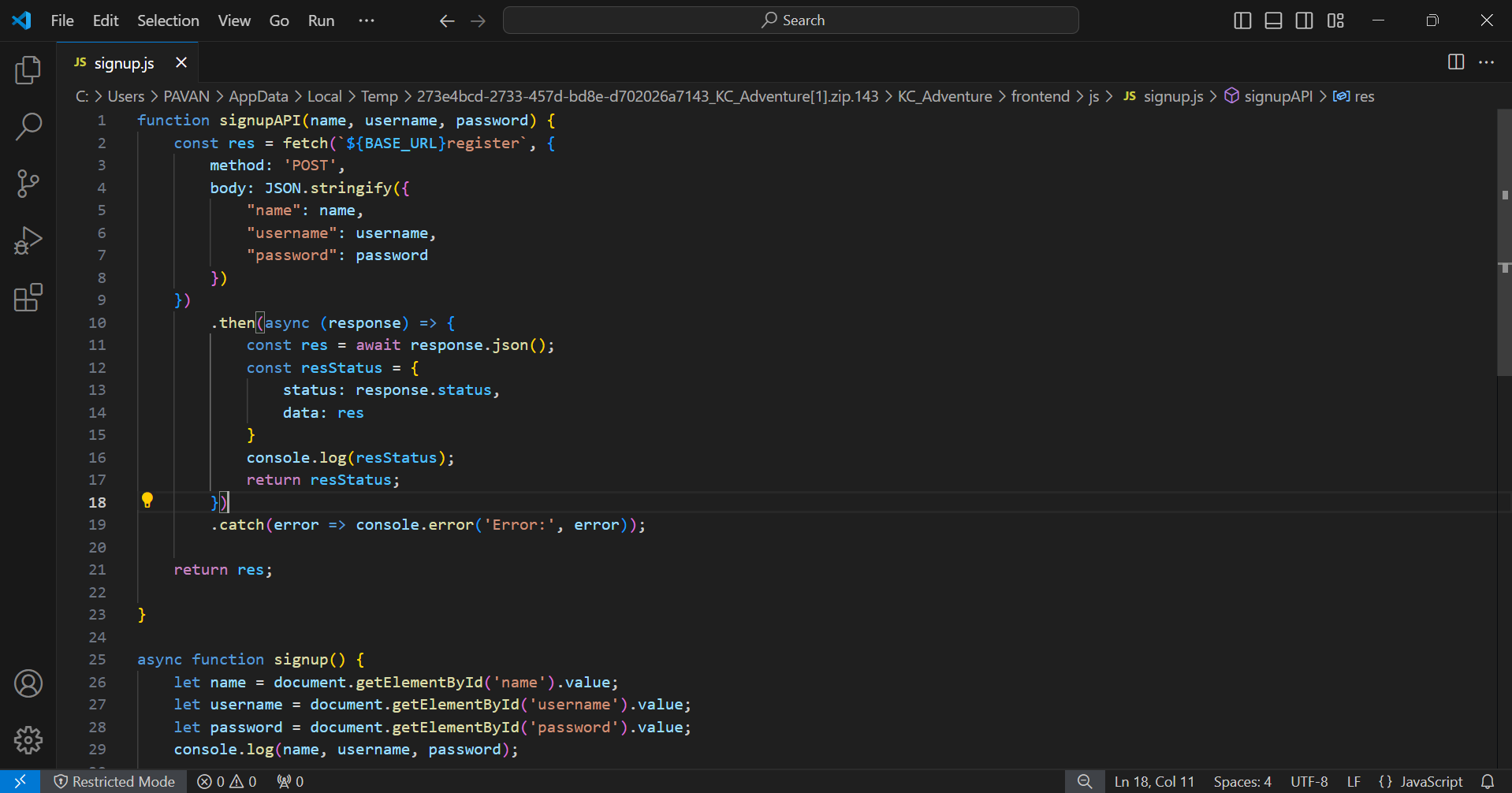
1. **Signup Screen**

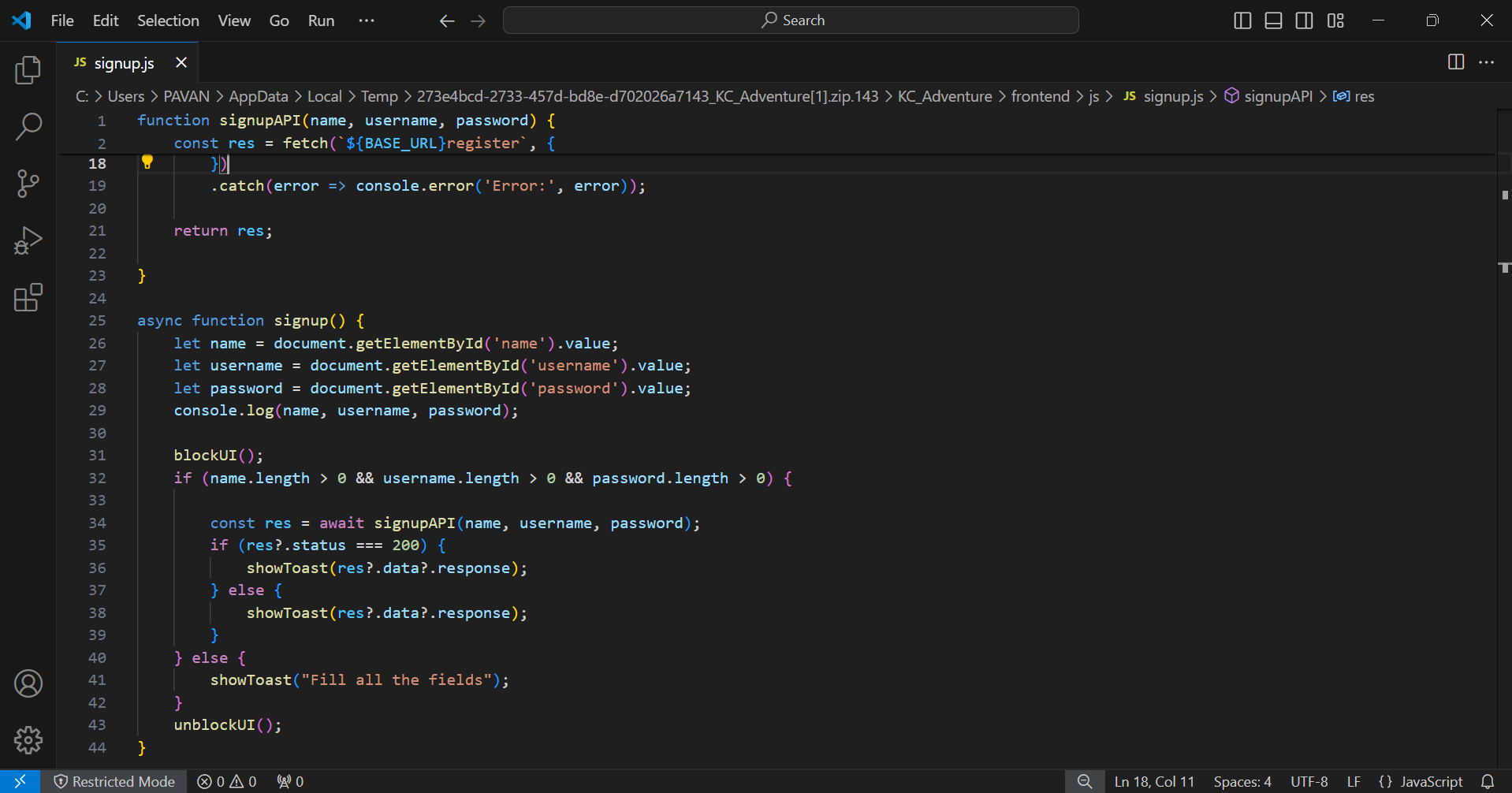
In this Signup Screen, we have the fields like full name, username, and password. Once these fields are filled, we need to hit the signup button then those details are stored into the database only if the details are not stored in the database previously. otherwise, we will get a suggestion like “Already have an account? Please Login”.

A screenshot of a login page

Description automatically generated

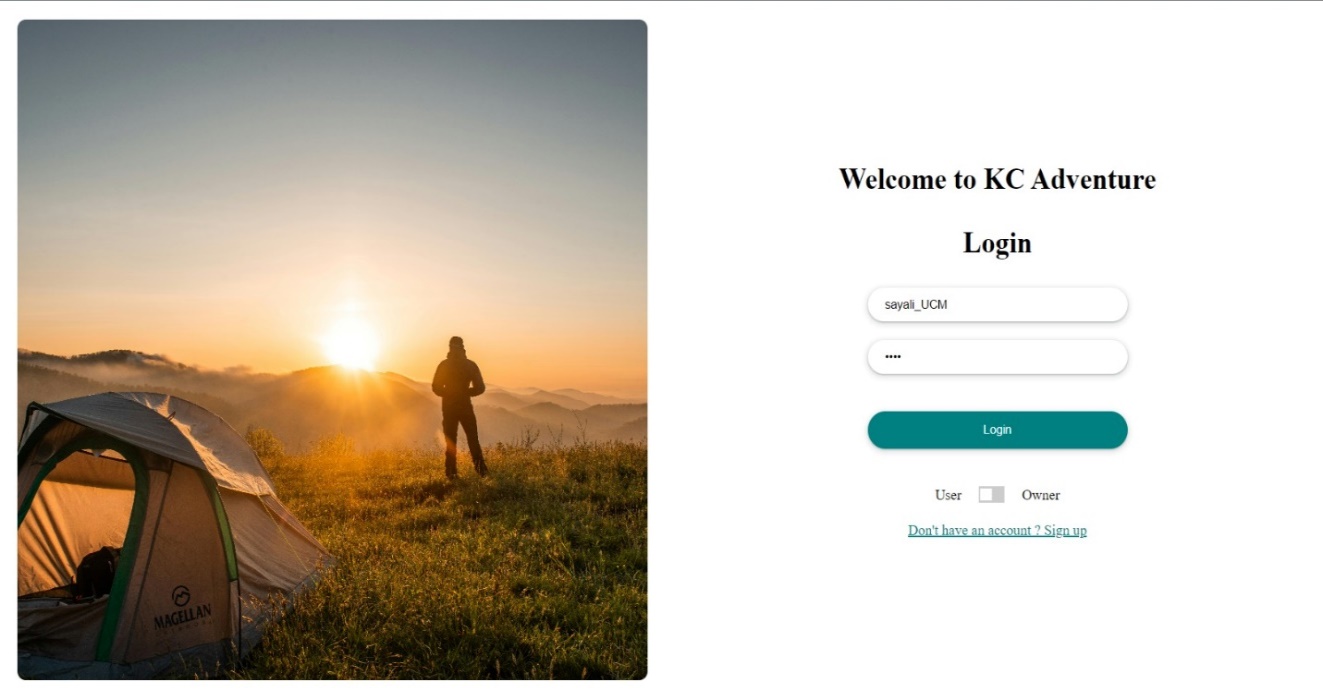
**Code Snippet:**



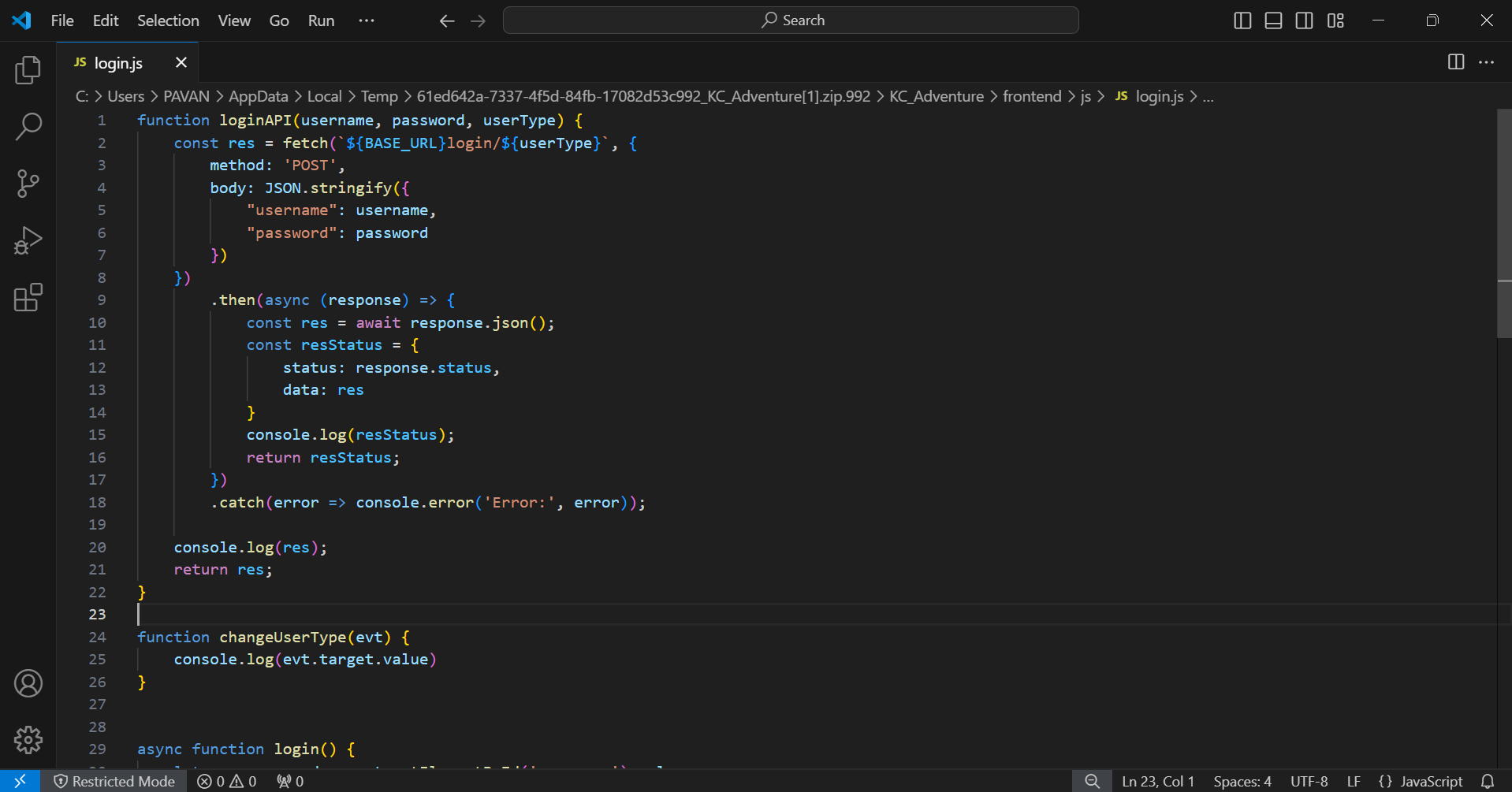


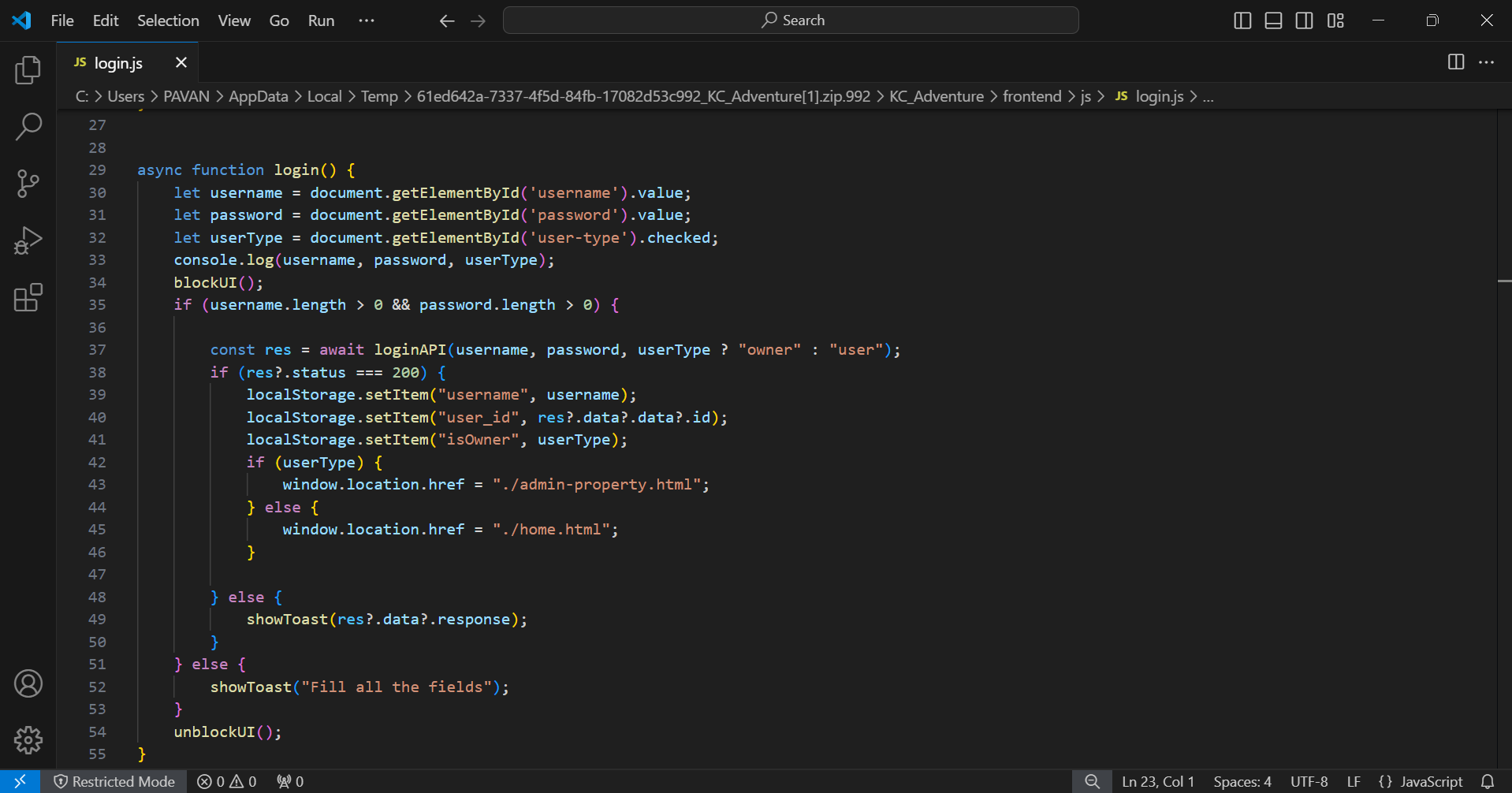
1. **Login Screen**

When we enter the username and password and clicks the login button, then the app checks if any user exist with those credentials (username and password) or not. If exists, then the user will be logged in to the Camp sites application or else, we have the suggestion like “Don’t have an account? Please Signup”. In this screen, we also have the toggle button to log in as a user or an owner.



**Code Snippet**





1. **Owner Home Screen**

Once we log in as an Owner into the Camp sites app, the owner will have access to update the existing properties details in camp properties page and camp bookings page. In the camp properties page, all the properties of the owner will be shown. Besides that, the owner has one more option like add new properties into the app or delete them. All these changes can be done anytime by the owner.

A screenshot of a web page

Description automatically generated

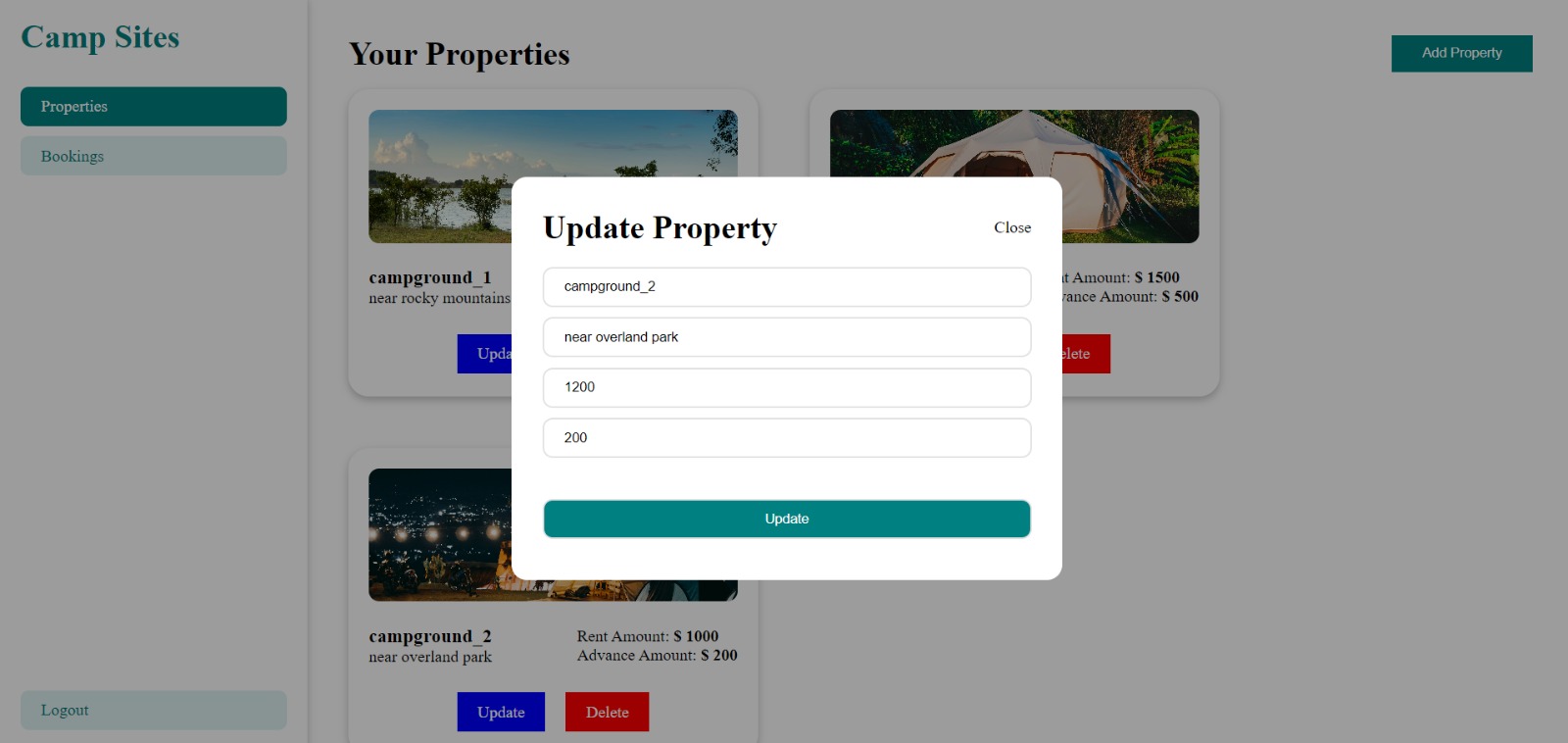
**Code Snippet:**

A screenshot of a computer

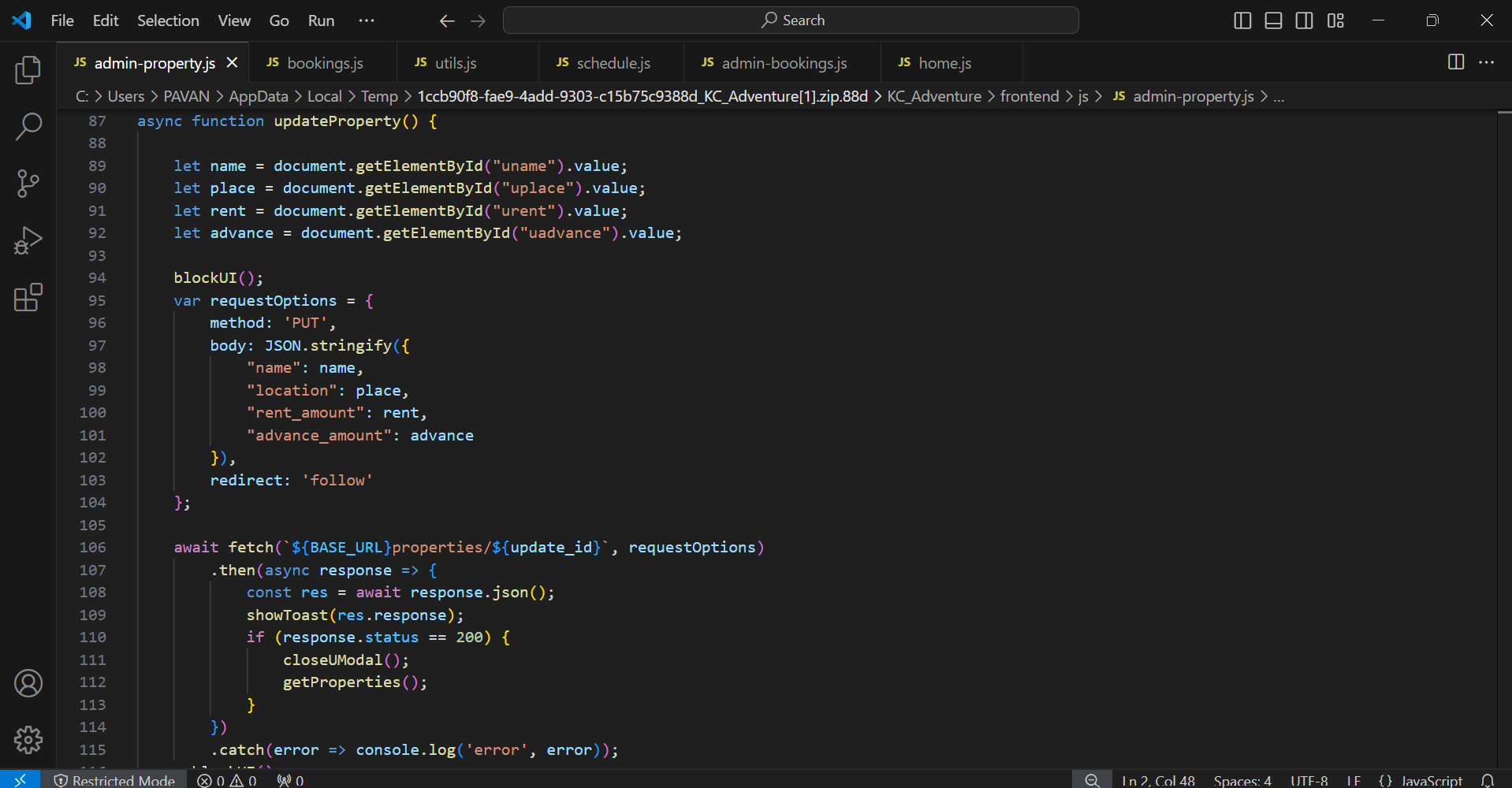
Description automatically generated

1. **Property Update Screen**

This update screen page is one of the forward pages for the home screen page for owner. As we mentioned earlier in this update property page, the owner can update the existing camp properties details like property name, its location, rent amount, and advance amount.



**Code Snippet:**



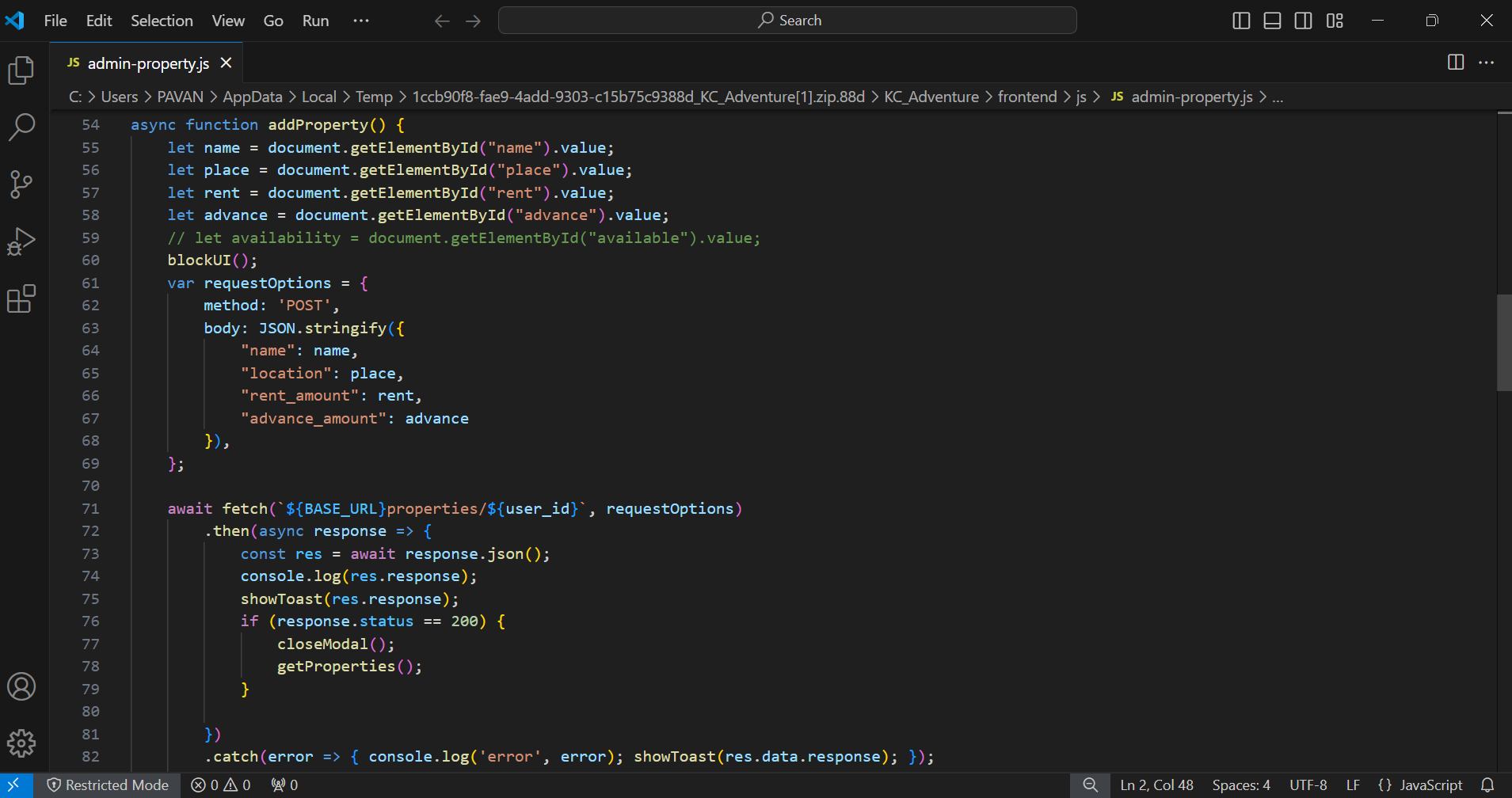
1. **Property Add Screen**

This add property page is also one of the forward pages for the home screen page for owner. The owner can add any camp property with details like property name, location, rent amount, and advance amount by clicking the update button.

A screenshot of a computer

Description automatically generated

**Code Snippet:**



1. **Bookings Screen**

In this camp bookings page, the owner can check for all the booking requests from the customers for their camp properties. Owner can both the confirm requests and reject.

A screenshot of a web page

Description automatically generated

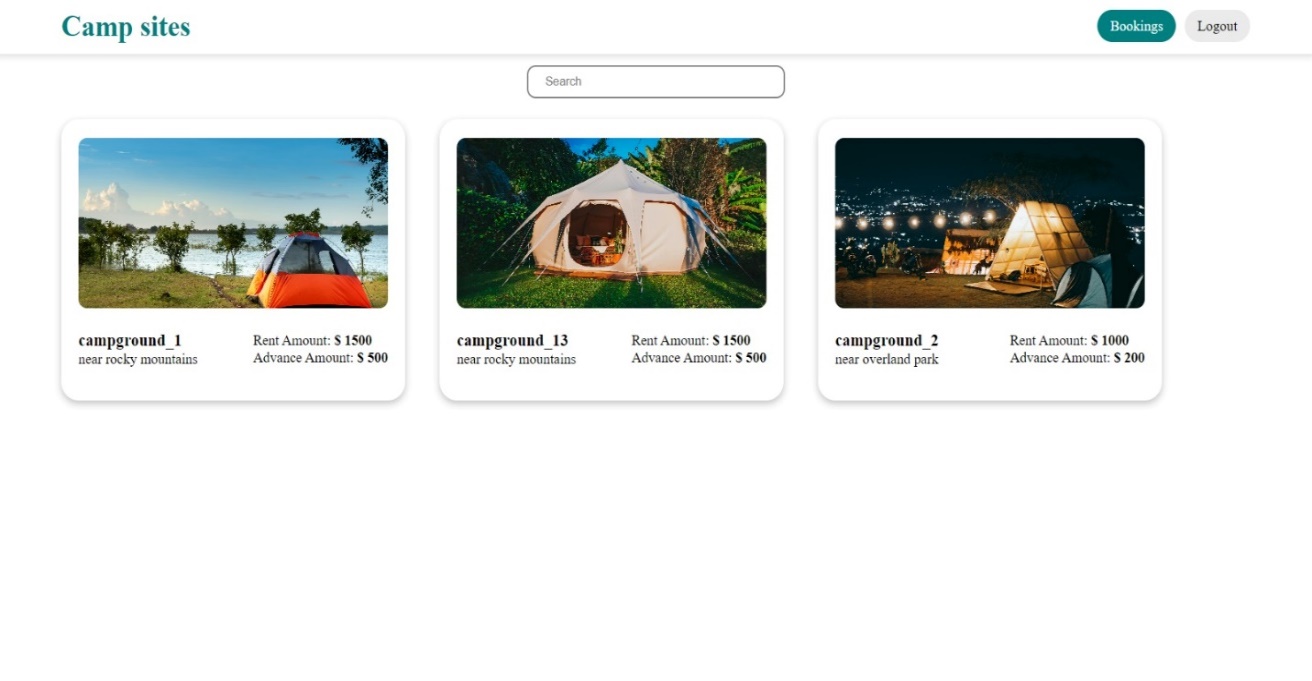
**Code Snippet:**

A screenshot of a computer

Description automatically generated

1. **User Home Screen**

In this home page like when we login as a user, A user can check all the camp properties listed by all the owners and can search for properties based on name, location, and cost.



**Code Snippet:**

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

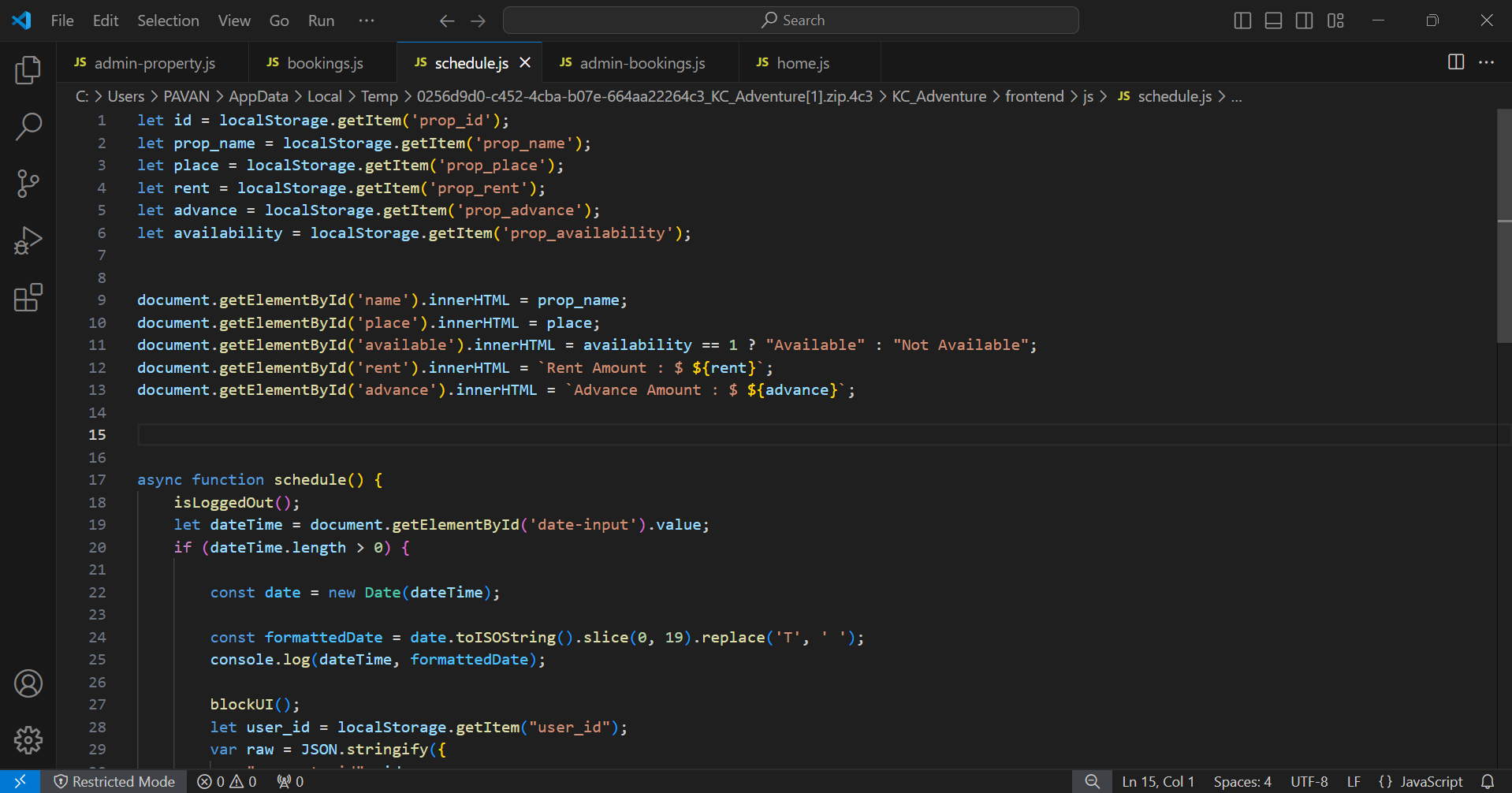
1. **Booking Screen**

In this bookings page, when the user satisfies with the camp properties details like location, rent amount, and advance amount then the user can make booking request based on their convenience.

A group of people sitting on the ground

Description automatically generated

**Code Snippet:**



1. **Booking History Screen**

In this booking history page, An user can check for his/her booking status and check for all their booking history from the past bookings to the future upcoming bookings.



**Code Snippet:**

A screenshot of a computer program

Description automatically generated