Main Objective

The main objective of this project is to create a comprehensive web application that allows users to plan trips by browsing some photos and checking weather information. The application integrates a weather API and a WebSocket server to provide real-time updates and image suggestions from the Pexels API.

Functionalities and Features

1. Users can input a city to get the current weather information including temperature, humidity, and description.
2. Weather data is fetched from the OpenWeatherMap API.

Trip Planning:

1. Users can fill out a form to plan their trip by providing details like destination, departure and return dates, budget, number of travelers, accommodation preferences, and departure time.
2. A trip receipt is generated displaying the provided trip details.

Image Search:

1. Users can search for images related to their trip destination using the Pexels API.
2. Retrieved images are displayed on the page.

WebSocket Integration:

1. Real-time communication with a WebSocket server to handle messages and updates related to weather data and image searches.
2. Automatic reconnection to the WebSocket server if the connection is lost.

Program Structure

HTML Structure:

1. The HTML structure consists of different sections including home, search, plan trip, help desk, and receipt sections.
2. Navigation links allow users to switch between these sections.

CSS:

1. CSS is used to style the various components and sections of the application.
2. The styling includes layout adjustments, button styling, form styling, and image display.

JavaScript:

1. JavaScript handles form submissions, API requests, WebSocket communication, and dynamic content updates.

WebSocket:

WebSocket Connection:

1. Establishes a WebSocket connection to the server for real-time communication.
2. Handles automatic reconnection if the WebSocket connection is lost.

WebSocket Event Handlers:

1. onopen: Logs the connection establishment.
2. onerror: Logs any errors that occur during the WebSocket communication.
3. onclose: Handles the closure of the WebSocket connection and attempts to reconnect after a delay.
4. onmessage: Processes incoming messages from the WebSocket server, handling weather data, image suggestions, and messages from other clients.

Integration with Other Functionalities:

1. Sends weather data and image search results to the WebSocket server to facilitate real-time updates.
2. Receives and displays data from the WebSocket server dynamically, ensuring the application stays up-to-date with the latest information.

Code Documentation

HTML:

```html
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Travel Planner</title>
<link rel="stylesheet" href="style.css">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.2.0/css/all.min.css">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta2/css/all.min.css"
```

```html
<footer>
    <p>&copy; 2024 Travel Planner. All rights reserved.</p>
</footer>
<script src="script.js"></script>
```

```html
<header>
    <nav class="navbar">
        <div class="nav-left">
            <a href="#" class="logo"><img src="resources/logo.png" alt="Logo"></a>
        </div>
        <input type="checkbox" id="check">
        <span class="menu" id="nav-links">
            <li><a href="#" id="home-link">Home</a></li>
            <li><a href="#" id="search-link">Browse</a></li>
            <li><a href="#" id="plan-trip-link">Plan Trip</a></li>
            <li><a href="#" id="help-desk-link">Help Desk</a></li>
            <label for="check" class="close-menu"><i class="fas fa-times"></i></label>
        </span>
        <label for="check" class="open-menu"><i class="fas fa-bars"></i></label>
    </nav>
</header>
```

The meta charset="UTF-8" tag ensures that the document is encoded using UTF-8, allowing for proper rendering of characters.

The <title> element sets the title of the webpage to "Travel Planner".

The <link> elements reference external CSS files to apply styles to the webpage and include Font Awesome icons.

The <footer> section contains a paragraph displaying copyright information for the website.

The <script> tag links an external JavaScript file named "script.js" to the webpage for dynamic functionality.

div element in the code is a generic container used to group HTML elements together.

class attribute is used to assign one or more class names to an HTML element. Classes are used to apply CSS styles or JavaScript behaviors to multiple elements with the same class name.

section elements in our HTML are used to define different sections of the webpage, such as the home section, search section, plan trip section, etc.

CSS:

```css
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600&display=swap');

* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}

body {
    font-family: "Poppins", sans-serif;
    background-color: #F5F5DC;
    overflow-y: hidden;
}

header {
    background-color: rgb(230, 188, 133);
    padding: 1em 0;
    text-align: center;
    z-index: 3;
}

.navbar {
    width: 100%;
    display: flex;
    justify-content: space-between;
    align-items: center;
    list-style: none;
    position: relative;
    background-color: rgb(230, 188, 133);
    padding: 8px 10px;
}
```

@import url - imports the "Poppins" font family from Google Fonts for use in the stylesheet.

* { } - Applies universal box-sizing, removes default margins and padding from all elements to create a consistent baseline for layout.

body { } - Sets the font to "Poppins", applies a beige background color, and hides the vertical scrollbar.

header { }- Styles the header with a specific background color, padding, centered text, and a higher z-index.

.navbar { } - Creates a flexible navigation bar with justified content, specific padding, and background color.

```css
.nav-left {
    align-items: center;
}

.logo img {
    width: 40px;
}

.menu {
    display: flex;
    padding-right: 20px;
}

.menu li {
    padding-left: 30px;
}

.menu li a {
    display: inline-block;
    text-decoration: none;
    color: white;
    text-align: center;
    transition: 0.15s ease-in-out;
    position: relative;
    text-transform: uppercase;
}

.menu li a::after {
    content: "";
    position: absolute;
    bottom: 0;
    left: 0;
    width: 0;
    height: 1px;
    background-color: white;
    transition: 0.15s ease-in-out;
}
.menu li a:hover::after {
    width: 100%;
}

.open-menu, .close-menu {
    position: absolute;
    color: white;
    cursor: pointer;
    font-size: 1.5rem;
    display: none;
}

.open-menu {
    top: 50%;
    right: 20px;
    transform: translateY(-50%);
}

.close-menu {
    top: 20px;
    right: 20px;
}

#check {
    display: none;
}

@media (max-width: 768px) {
    .menu {
        flex-direction: column;
        align-items: center;
        justify-content: center;
        width: 80%;
        height: 100vh;
        position: fixed;
        top: 0;
    .menu li {
        margin-top: 40px;
    }

    .menu li a {
        padding: 10px;
    }

    .open-menu, .close-menu {
        display: block;
    }

    #check:checked ~ .menu {
        right: 0;
    }
}

.section {
    display: none;
    padding: 2em;
    text-align: center;
}

.section.active {
    display: block;
}

.welcome_traveler {
    position: absolute;
    top: 20%;
    left: 60%;
    transform: translateX(-50%);
    font-size: 60px;
    color: rgb(255, 255, 255);
@media (max-width: 768px) {
    .menu {
        flex-direction: column;
        align-items: center;
        justify-content: center;
        width: 80%;
        height: 100vh;
        position: fixed;
        top: 0;
        right: -100%;
        z-index: 100;
        background-color: rgb(230, 188, 133);
        transition: all 0.2s ease-in-out;
    }

    .menu li {
        margin-top: 40px;
    }

    .menu li a {
        padding: 10px;
    }

    .open-menu, .close-menu {
        display: block;
    }

    #check:checked ~ .menu {
        right: 0;
    }
}

.section {
    display: none;
    padding: 2em;
```

.nav-left { } - will aligns items in the left part of the navigation bar centrally.

.logo img {} - Sets the logo image width.

.menu { } - Displays the menu as a flexible container with right padding.

.menu li { } - Adds left padding to each list item.

.menu li a { } - Styles anchor links within menu items with no underline, white text color, center alignment, and a transition effect.

.menu li a::after { } - Adds an underline effect to menu links on hover.

.menu li a:hover::after { } - this expands the underline to full width on hover.

.open-menu, .close-menu {} - Styles the open and close menu icons for responsive design, including position, color, and size.

.open-menu {}- Positions the open menu icon.

.close-menu {} - Positions the close menu icon.

#check {} - Hides the checkbox used for toggling the menu in responsive design.

@media () {} - Adjusts the layout for screens smaller than 768px, including making the menu a fixed vertical list and displaying toggle icons.

.section { } - Hides sections by default and centers their text.

```
.container_1 {
    top: 20%;
}

.container_2 {
    top: 45%;
}

.container_3 {
    top: 70%;
}

.weatherForm{
    margin: 20px;
}

.cityInput{
    padding: 10px;
    font-size: 1.5rem;
    font-weight: bold;
    border: 2px solid hsla(0, 0%, 20%, 0.3);
    border-radius: 10px;
    margin: 10px;
    width: 300px;
}

button[type="submit"]{
    padding: 10px;
    font-weight: bold;
    font-size: 1.5rem;
    background-color: #D0B59A;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}
button[type="submit"]:hover{
    background-color: #D0B59A;
}

.card {
    background: linear-gradient(180deg, #D0B59A, #B5926D);
    padding: 50px;
    border-radius: 20px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    min-width: 300px;
    height: 300px;
    display: flex;
    flex-direction: column;
    align-items: center;
}

.card {
    background: linear-gradient(180deg, #D0B59A, #B5926D);
    padding: 20px;
    border-radius: 20px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    min-width: 300px;
    height: auto;
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
}

.card img {
    margin: 10px;
    height: 150px;
    width: auto;
    border-radius: 10px;
    object-fit: cover;
}
.section.active {
    display: block;
}

.welcome_traveler {
    position: absolute;
    top: 20%;
    left: 60%;
    transform: translateX(-50%);
    font-size: 60px;
    color: rgb(255, 255, 255);
    white-space: nowrap;

}

.slogan {
    position: absolute;
    top: 40%;
    left: 60%;
    transform: translate(-50%, -50%);
    font-size: 19px;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    color: white;

}

@media (max-width: 1200px) {
    .welcome_traveler {
        font-size: 50px;
    }
    .slogan {
        font-size: 18px;
    }
}
        font-size: 40px;
    }
    .slogan {
        font-size: 16px;
    }
}
@media (max-width: 768px) {
    .welcome_traveler {
        font-size: 30px;
    }
    .slogan {
        font-size: 14px;
    }
}
@media (max-width: 576px) {
    .welcome_traveler {
        font-size: 20px;
    }
    .slogan {
        font-size: 12px;
    }
}
.container {
    width: 20%;
    height: 5vh;
    position: absolute;
    overflow: hidden;
    top: 50%;
    transform: translateY(-50%);
}
```

.section.active {} - Makes active sections visible.

.welcome_traveler {} - Styles the welcome text with specific font size, color, and positioning.

.slogan {} - this styles the slogan text with specific font properties and positioning in our HTML.

@media {} - Adjusts font sizes of the welcome text and slogan for various screen sizes.

.container { }- Styles a container with specific dimensions and central alignment.

.container_1, .container_2, .container_3 {} - Positions containers at different vertical heights.

.weatherForm {} - Adds margin to the weather form.

.cityInput {} - Styles the city input field with padding, font size, border, and margin.

button{} - Styles the submit button with padding, font properties, background color, border, and cursor.

button {}- Styles the submit button on hover.

.card { } - Styles the card with a gradient background, padding, border radius, shadow, and flex properties.

.card img {} - Styles images within the card with margins, size, border radius, and object-fit.

```css
.cityDisplay, .tempDisplay{
    font-size: 2.5rem;
    font-weight: bold;
    color: hsla(0, 0%, 0%, 0.8);
    margin-bottom: 5px;
}

.humidityDisplay{
    font-weight: bold;
    margin-bottom: 25px;
    font-size: 1.5rem;
}

.descDisplay{
    font-style: italic;
    font-weight: bold;
    font-size: 1.5rem;
}

.errorDisplay{
    font-size: 1.5rem;
    color: hsla(0, 0%, 0%, 0.75);
}

.slides_1,
.slides_2,
.slides_3 {
    display: flex;
    transition: transform 1s ease;
    padding-top: 1%;
}

.slides_1 img,
.slides_2 img,
.slides_3 img {
```

```css
.video-container {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    overflow: hidden;
    z-index: -1;
}

video {
    position: absolute;
    top: 50%;
    left: 50%;
    min-width: 100%;
    min-height: 100%;
    width: auto;
    height: auto;
    transform: translate(-50%, -50%);
}

footer {
    background-color: #9f9f9f80;
    color: white;
    text-align: center;
    padding: 2px;
    position: fixed;
    width: 100%;
    bottom: 0;
}

#plan-trip-section {
```

```css
#trip-form {
    display: flex;
    flex-direction: column;
    gap: 10px;
    max-width: 500px;
    margin: 0 auto;
}

#trip-form div {
    flex: 1 1 auto;
}

#trip-form label {
    display: block;
    margin-bottom: 5px;
}

#trip-form input[type="text"],
#trip-form input[type="date"],
#trip-form input[type="number"],
#trip-form select {
    width: 100%;
    padding: 8px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

#trip-form input[type="range"] {
    width: 100%;
}

#budget-value {
    display: inline-block;
```

```css
    display: inline-block;
    margin-left: 10px;
}

#trip-form button[type="submit"] {
    padding: 5px 5px;
    background-color: #D0B59A;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background-color: #D0B59A;
}
```

.cityDisplay, .tempDisplay { } - Styles city and temperature information with specific font properties.

.humidityDisplay { } - Styles humidity information with specific font properties.

.descDisplay { } - Styles description information with specific font properties.

.errorDisplay { } - Styles the error display with specific font properties.

.slides_1, .slides_2, .slides_3 { } - Controls the appearance and transitions of image slides.

.container_1: - Applies hover effects to containers with slides.

.video-container {} - Styles the video container to be full screen and behind other elements.

video {} - Styles the video to cover the full container area.

footer {} - Styles the footer with a background color, white text, centered alignment, padding, and fixed position at the bottom.

#plan-trip-section {}- Styles the trip planning section with padding

#plan-trip-form {} - Styles the trip planning form with flexible column layout and centered alignment.

#plan-trip-form label { } - Styles the form labels with bold text, margins, font size, and white color.

#plan-trip-form input {} - Styles the input fields with padding, font size, border, and margin.

#plan-trip-form select {} - Styles the select dropdowns with padding, font size, border, and margin.

#plan-trip-form button {} - Styles the submit button similarly to the weather form button, with padding, font properties, background color, border, and cursor.

#plan-trip-form button:hover {} - Styles the button on hover.

```css
#receipt h3 {
    font-size: 20px;
}

#receipt p {
    margin: 5px 0;
}

#receipt strong {
    font-weight: bold;
}

#receipt button {
    background-color: #D0B59A;
    color: #fff;
    border-radius: 5px;
    padding: 10px 20px;
    margin-top: 20px;
    cursor: pointer;
}

#receipt button:hover {
    background-color: #D0B59A;
}

.Q1,
.Q2,
.Q3,
.Q4
.Q5{
    border: 2px solid #000;
    padding: 10px;
    border-radius: 20px;
    margin: 22px;
    z-index: 1;
}
.Q2,
.Q3,
.Q4
.Q5{
    border: 2px solid #000;
    padding: 10px;
    border-radius: 20px;
    margin: 22px;
    z-index: 1;
}

.hidden {
    display: none;
}

.card_place {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    max-height: 400px;
    overflow-y: auto;
}

.card_place img {
    width: 200px;
    height: 200px;
    object-fit: cover;
    margin: 10px;
}
.container {
    max-width: 800px;
    margin: 50px auto;
    padding: 20px;
    background-color: white;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    text-align: center;
}

h3 {
    margin-bottom: 20px;
}

form {
    display: flex;
    justify-content: center;
    margin-bottom: 20px;
}

#search-input {
    width: 70%;
    padding: 10px;
    font-size: 16px;
    border: 1px solid #ccc;
    border-radius: 4px;
}


.patterns {
    width: 45%;
    float: left;
    margin-top: 13%;
    height: 55%;
    flex-wrap: wrap;
}
```

#receipt p {} - Adds vertical margins of 5px to paragraphs inside the receipt element.

#receipt strong {} - Makes text within `<strong>` tags inside the receipt element bold.

#receipt button {} - Styles buttons within the receipt element with a specific background color, white text, rounded corners, padding, a top margin, and a pointer cursor.

#receipt button:hover {} - Changes the background color of buttons on hover.

.Q1, .Q2, .Q3, .Q4, .Q5 { } - Styles these elements with a solid black border, padding, rounded corners, margin, and a specific z-index.

.hidden {} - Hides elements with the hidden class.

.card_place {} - Styles the card container to display flexibly, wrap content, center items, and allow vertical scrolling.

.card_place img {} - Styles images inside the card container with specific dimensions, object-fit, and margins.

.container {} - Styles the container with a max width, auto margins for centering, padding, background color, box shadow, and centered text.

h3 {} - Adds a bottom margin to `<h3>` headers.

form {} - Styles forms to display flexibly, center content, and add a bottom margin.

#search-input {} - Styles the search input field with width, padding, font size, border, and rounded corners.

.patterns {} - Styles the patterns container with specific dimensions, float property, margin, and wrapping content.

```css
svg text {
    font-family: Lora;
    letter-spacing: 5px;
    stroke: #4b3b42;
    font-size: 100px;
    font-weight: 700;
    stroke-width: 3;
    animation: textAnimate 3s infinite alternate;
}

@keyframes textAnimate {
    0% {
        stroke-dasharray: 0 50%;
        stroke-dashoffset: 20%;
        fill: #9c8481;
    }
    100% {
        stroke-dasharray: 50% 0;
        stroke-dashoffset: -20%;
        fill: #fff0db;
    }
}

.content {
    padding: 10px;
}

details {
    border: 1px solid #2c2b2b;
    border-radius: 4px;
    padding: 10px;
    margin: 5px;
    width: 45%;
    margin-left: 50%;
    background-color: rgb(230, 188, 133);
    font-family: Arial, Helvetica, sans-serif;
    box-shadow: 3px 2px #db9393a7;
}

summary {
    font-weight: bold;
    margin: -0.5em -0.5em;
    padding: 1em;
    cursor: pointer;
}

summary:hover {
    background: #7b6e6e4c;
}

@media (max-width: 700px) {
    .patterns {
        width: 100%;
        float: none;
        margin-top: 0;
        height: auto;

    }

    svg text {
        font-size: 50px;
    }

    details {
        width: 100%;
        margin-left: 0;
    }
}

* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
}

.wrapper {
    display: flex;
    flex-direction: column;
    gap: 20px;
    max-width: 100%;
}

.row {
    display: flex;
    align-items: center;
    position: relative;
    margin-bottom: 20px;
}

.row i {
    top: 50%;
    height: 44px;
    width: 44px;
    color: #343F4F;
    cursor: pointer;
    font-size: 1.15rem;
    position: absolute;
    text-align: center;
    line-height: 44px;
    background: #fff;
    border-radius: 50%;
    transform: translateY(-50%);
    transition: transform 0.1s linear;
}

.row i:active {
    transform: translateY(-50%) scale(0.9);
}

.row i:hover {
    background: #f2f2f2;
}
```

svg text {} - Styles text with a specific font, letter spacing, stroke properties, font size, and animation.

@keyframes textAnimate {} - Defines a keyframe animation for SVG text stroke and fill properties.

.content {} - Adds padding to content elements.

details {} - elements with a border, padding, margins, background color, font, and box shadow.

summary {}- elements with bold text, margins, padding, and a pointer cursor.

summary:hover {} - Changes the background of `<summary>` elements on hover.

@media (max-width: 700px) {}- Adjusts the layout for screens narrower than 700px, including resizing patterns and SVG text, and full-width `<details>` elements.

* {} - Applies universal box-sizing, margin, padding, and font family for all elements.

.wrapper {} - Styles the wrapper container with flexible column layout and gaps between items.

.row {} - Styles rows with flexible alignment, relative positioning, and bottom margin.

.row i {} - Styles icons within rows with specific positioning, dimensions, colors, cursor, font size, and transition effects.

.row i:active {} - Scales icons when active.

.row i:hover {} - this changes the background of icons on hover.

```css
.row i:first-child {
    left: -22px;
    display: none;
}

.row i:last-child {
    right: -22px;
}

.carousel {
    font-size: 0px;
    cursor: pointer;
    overflow: hidden;
    white-space: nowrap;
    scroll-behavior: smooth;
}

.carousel.dragging {
    cursor: grab;
    scroll-behavior: auto;
}

.carousel.dragging img {
    pointer-events: none;
}


.image-container {
    position: relative;
    display: inline-block;
    margin-right: 10px;
}

.image-container img {
    width: 500px;
    height: 300px;
    object-fit: cover;
```

```css
.image-container img {
    width: 500px;
    height: 300px;
    object-fit: cover;
    border-radius: 8px;
    user-select: none;
}

.image-container img:first-child {
    margin-left: 0px;
}


.image-container i.fa-trophy {
    position: absolute;
    top: 25px;
    right: 10px;
    color: gold;
    font-size: 24px;
}

#left-1 {
  position: absolute;
  top: 50%;
  left: 10px;
  transform: translateY(-50%);
  z-index: 1;
  cursor: pointer;
}

.image-container i.fa-heart {
  position: absolute;
  top: 25px;
  right: 10px;
  color: black;
  font-size: 24px;
}
```

```css
#home-section {
    height: 100vh;
    overflow-y: scroll;
}


.scrollable-content {
    max-height: calc(100vh - 200px);
    overflow-y: auto;
}


.video-container {
    max-height: 100%;
    overflow: hidden;
}


.heading {
    color: white;
}
```

.row i:first-child {} - Hides the first icon within rows.

.row i:last-child {}` - Positions the last icon within rows.

.carousel {} - Styles the carousel with zero font size, cursor, overflow, and smooth scrolling.

.carousel.dragging {} - Changes the cursor and scrolling behavior when dragging the carousel.

.carousel.dragging img {} - Disables pointer events for images when dragging.

.image-container {} - Styles image containers with relative positioning, display properties, and margins.

.image-container img {} - Styles images within containers with specific dimensions, object-fit, and border radius.

.image-container i.fa-trophy {} - Positions the trophy icon within image containers.

#left-1 {} - Positions the left navigation button within image containers.

.image-container i.fa-heart {} - Positions the heart icon within image containers.

#home-section {} - Styles the home section with full viewport height and vertical scrolling.

.scrollable-content {} - Styles scrollable content with max height and vertical overflow.

.video-container {} - Styles the video container with max height and overflow properties.

.heading { } - Styles headings with white color.

JavaScript:

```
const weatherForm = document.querySelector(".weatherForm");
const cityInput = document.querySelector(".cityInput");
const card = document.querySelector(".card");
const apiKey = "0ea5fed713a4fcb113af98dac3e011c8";
let ws;

function connectWebSocket() {
    ws = new WebSocket('ws://localhost:3000');

    ws.onopen = function () {
        console.log('Connected to WebSocket server.');
    };

    ws.onerror = function (error) {
        console.error('WebSocket error:', error);
    };

    ws.onclose = function () {
        console.log('WebSocket connection closed. Reconnecting...');
        setTimeout(connectWebSocket, 1000);
    };

    ws.onmessage = function (event) {
        try {
            const parsedMessage = JSON.parse(event.data);
            if (parsedMessage.source === 'pexels') {
                displayImages(JSON.parse(parsedMessage.data));
            } else if (parsedMessage.source === 'weather') {
                displayWeatherInfo(JSON.parse(parsedMessage.data));
            } else {
                console.log(`Received message from client: ${parsedMessage}`);
            }
        } catch (error) {
            console.error('Error parsing message:', error);
        }
connectWebSocket();

function sendTestMessage() {
    if (ws.readyState === WebSocket.OPEN) {
        ws.send('Test message from client');
    } else {
        console.error('WebSocket is not open. ReadyState: ' + ws.readyState);
    }
}

document.addEventListener('DOMContentLoaded', function() {
    let hidePlanTrip = false;

    const links = {
        'home-link': 'home-section',
        'search-link': 'search-section',
        'plan-trip-link': 'plan-trip-section',
        'help-desk-link': 'help-desk-section'
    };

    for (const [linkId, sectionId] of Object.entries(links)) {
        document.getElementById(linkId).addEventListener('click', function(event) {
            event.preventDefault();
            showSection(sectionId);
        });
    }

    function showSection(sectionId) {
        document.querySelectorAll('.section').forEach(section => {
            section.classList.remove('active');
        });
        document.getElementById(sectionId).classList.add('active');
        if (sectionId === 'plan-trip-section') {
            document.getElementById('receipt-section').classList.add('active');
            if (hidePlanTrip) {
                document.getElementById('plan-trip-section').classList.remove('active');
async function searchImages(query) {
    const apiKey = '03ommpMyMFF7QnUYim8fL3isJ3dPqbgYc0cWZfrFQywUuYLNC2DTgzft';
    const apiUrl = `https://api.pexels.com/v1/search?query=${query}`;

    const response = await fetch(apiUrl, {
        headers: {
            Authorization: apiKey
        }
    });

    if (!response.ok) {
        throw new Error('Could not fetch image data');
    }

    const data = await response.json();
    return data.photos;
}

const searchForm = document.getElementById('search-form');
searchForm.addEventListener('submit', async function(event) {
    event.preventDefault();

    const query = document.getElementById('search-input').value.trim();
    if (query) {
        try {
            const images = await searchImages(query);
            displayImages(images);
            sendMessageToWebSocket(JSON.stringify(images));
        } catch (error) {
            console.error(error);
            displayError(error.message);
        }
    } else {
        displayError('Please enter a search query');
function displayImages(images) {
    const card = document.querySelector('.card_place');
    card.innerHTML = '';
    card.style.display = 'flex';

    if (images.length === 0) {
        displayError('No images found');
        return;
    }

    images.forEach(image => {
        const img = document.createElement('img');
        img.src = image.src.medium;
        img.alt = image.alt;
        card.appendChild(img);
    });
}
```

const weatherForm - This variable selects the weather form element from the HTML document.

const cityInput - This variable selects the city input field from the HTML document.

const card - This variable selects the card element from the HTML document.

const apiKey - This variable holds the API key required for accessing weather data.

let ws - This variable is declared to hold the WebSocket connection.

function connectWebSocket - This function establishes a WebSocket connection with the server, handles connection errors, and attempts to reconnect in case of connection closure.

function sendTestMessage - This function sends a test message to the WebSocket server if the connection is open.

document.addEventListener('DOMContentLoaded', function - This event listener executes JavaScript code once the DOM content is loaded. It sets up event listeners for navigation links, handles form submissions, and displays trip details.

async function searchImages(query) - This asynchronous function fetches images related to a search query from the Pexels API.

function displayImages(images) - This function displays the fetched images on the webpage.

```
    function displayError(message) {
        const card = document.querySelector('.card');
        card.innerHTML = '';
        card.style.display = 'flex';

        const errorDisplay = document.createElement('p');
        errorDisplay.textContent = message;
        errorDisplay.classList.add('errorDisplay');
        card.appendChild(errorDisplay);
    }

    function sendMessageToWebSocket(message) {
        if (ws.readyState === WebSocket.OPEN) {
            ws.send(message);
        } else {
            console.error('WebSocket is not open. ReadyState: ' + ws.readyState);
        }
    }

weatherForm.addEventListener("submit", async event => {
    event.preventDefault();

    const city = cityInput.value;

    if (city) {
        try {
            const weatherData = await getWeatherData(city);
            if (ws.readyState === WebSocket.OPEN) {
                ws.send(JSON.stringify(weatherData));
            } else {
                console.error('WebSocket is not open. ReadyState: ' + ws.readyState);
            }
            displayWeatherInfo(weatherData);
        } catch (error) {
            console.error(error);
async function getWeatherData(city) {
    const apiUrl = `https://api.openweathermap.org/data/2.5/weather?q=${city},PH&appid=${apiKey}`;

    const response = await fetch(apiUrl);

    if (!response.ok) {
        throw new Error("Could not fetch weather data");
    }

    return await response.json();
}

function displayWeatherInfo(data) {
    const { name: city,
        main: { temp, humidity },
        weather: [{ description }] } = data;

    card.textContent = "";
    card.style.display = "flex";

    const cityDisplay = document.createElement("h1");
    const tempDisplay = document.createElement("p");
    const humidityDisplay = document.createElement("p");
    const descDisplay = document.createElement("p");

    cityDisplay.textContent = city;
    tempDisplay.textContent = `${(temp - 273.15).toFixed(1)}°C`;
    humidityDisplay.textContent = `Humidity: ${humidity}%`;
    descDisplay.textContent = description;

    cityDisplay.classList.add("cityDisplay");
    tempDisplay.classList.add("tempDisplay");
    humidityDisplay.classList.add("humidityDisplay");
    descDisplay.classList.add("descDisplay");
function displayError(message) {
    const errorDisplay = document.createElement("p");
    errorDisplay.textContent = message;
    errorDisplay.classList.add("errorDisplay");

    card.textContent = "";
    card.style.display = "flex";
    card.appendChild(errorDisplay);
}

    let slideIndex = 0;
    showSlides();

    function showSlides() {
        let slides1 = document.querySelectorAll('.slides_1 img');
        let slides2 = document.querySelectorAll('.slides_2 img');
        let slides3 = document.querySelectorAll('.slides_3 img');

        for (let i = 0; i < slides1.length; i++) {
            slides1[i].style.display = "none";
        }
        for (let i = 0; i < slides2.length; i++) {
            slides2[i].style.display = "none";
        }
        for (let i = 0; i < slides3.length; i++) {
            slides3[i].style.display = "none";
        }

        slideIndex++;
        if (slideIndex > slides1.length) { slideIndex = 1; }

        slides1[slideIndex-1].style.display = "block";
        slides2[slideIndex-1].style.display = "block";
        slides3[slideIndex-1].style.display = "block";
const carousels = document.querySelectorAll(".carousel"),
    arrowIcons = document.querySelectorAll(".row i");

let isDragStart = false, isDragging = false, prevPageX, prevScrollLeft, positionDiff;

const showHideIcons = (carousel, icons) => {
    let scrollWidth = carousel.scrollWidth - carousel.clientWidth;
    icons[0].style.display = carousel.scrollLeft == 0 ? "none" : "block";
    icons[1].style.display = carousel.scrollLeft == scrollWidth ? "none" : "block";
}

arrowIcons.forEach((icon, i) => {
    icon.addEventListener("click", () => {
        const row = icon.parentElement;
        const carousel = row.querySelector(".carousel");
        const icons = row.querySelectorAll("i");
        let firstImgWidth = carousel.querySelector("img").clientWidth + 14;
        carousel.scrollLeft += icon.id.includes("left") ? -firstImgWidth : firstImgWidth;
        setTimeout(() => showHideIcons(carousel, icons), 60);
    });
});

const autoSlide = (carousel, firstImgWidth) => {
    if (carousel.scrollLeft - (carousel.scrollWidth - carousel.clientWidth) > -1 || carousel.scrollLeft <= 0) return;
    positionDiff = Math.abs(positionDiff);
    let valDifference = firstImgWidth - positionDiff;
    if (carousel.scrollLeft > prevScrollLeft) {
        return carousel.scrollLeft += positionDiff > firstImgWidth / 3 ? valDifference : -positionDiff;
    }
    carousel.scrollLeft -= positionDiff > firstImgWidth / 3 ? valDifference : -positionDiff;
}

const dragStart = (e, carousel) => {
    isDragStart = true;
    prevPageX = e.pageX || e.touches[0].pageX;
    prevScrollLeft = carousel.scrollLeft;
```

function displayError(message) - This function displays error messages on the webpage.

function sendMessageToWebSocket(message) - This function sends a message to the WebSocket server.

weatherForm.addEventListener("submit", async event => - This event listener handles the submission of the weather form, fetches weather data based on the entered city, and displays the weather information on the webpage.

async function getWeatherData(city) - This asynchronous function fetches weather data from the OpenWeatherMap API based on the entered city.

function displayWeatherInfo(data) - This function displays the weather information on the webpage.

function displayError(message) - This function displays error messages related to weather data retrieval on the webpage.

function showSlides() - This function implements a slideshow by cycling through images at regular intervals.

const details = document.querySelectorAll('details'); - This variable selects all `<details>` elements from the HTML document.

const carousels = document.querySelectorAll(".carousel"), arrowIcons = document.querySelectorAll(".row i");` - These variables select carousel elements and arrow icons for image scrolling.

WebSocket:

```
const express = require('express');
const http = require('http');
const WebSocket = require('ws');

const app = express();
const server = http.createServer(app);
const wss = new WebSocket.Server({ server });

wss.on('connection', (ws) => {
    console.log('Client connected');

    ws.on('message', (message) => {
        try {
            const parsedMessage = JSON.parse(message);
            if (parsedMessage.source === 'pexels') {
                handlePexelsMessage(parsedMessage.data);
            } else {
              console.log(`Received message from client: ${message}`);
            }
        } catch (error) {
            console.error('Error parsing message:', error);
        }
    });

    ws.on('close', () => {
        console.log('Client disconnected');
    });
});

function handlePexelsMessage(data) {
  console.log('Received message from Pexels API:', data);
}

server.listen(3000, () => {
    console.log("WebSocket server is listening on port 3000");
});
```

const express = require('express'); - This imports the Express framework to create a web server.

const http = require('http'); - This variable imports the HTTP module to create an HTTP server.

const WebSocket = require('ws'); - This variable imports the WebSocket library to enable WebSocket communication.

const app = express(); - This variable initializes the Express application.

const server = http.createServer(app); - This variable creates an HTTP server using the Express application.

const wss = new WebSocket.Server({ server }); - This variable creates a WebSocket server and attaches it to the HTTP server.

wss.on('connection', (ws) => {}); - This event listener handles new WebSocket client connections and logs a message when a client connects.

ws.on('message', (message) => {}); - This event listener handles incoming messages from WebSocket clients, parses the messages, and processes them based on their source.

function handlePexelsMessage(data) {}; - This function handles messages from the Pexels API by logging the received data.

ws.on('close', () => {}); - This event listener handles the disconnection of WebSocket clients and logs a message when a client disconnects.

server.listen(3000, () => { }); - This function starts the HTTP server and logs a message indicating that the WebSocket server is listening on port 3000.