**VISUAL INTERFACE DESIGN**

This document outlines a sophisticated design for a digital solution tailored to revolutionize user engagement in the tourism and leisure sector. Leveraging innovative technologies, the proposed system aims to enhance user experience, streamline operations, and ensure robust regulatory compliance. This documentation serves as a pivotal resource for technical and non-technical stakeholders, allowing for informed decision-making throughout the development process

1.) **User interface concepts**

Landing page:



The image above is the landing page a user sees when they first visit the site, the primarily focus is the two lionesses in the middle of the background which draws the user's attention with a context that says, “YOUR JOURNEY STARTS HERE.”

Homepage:

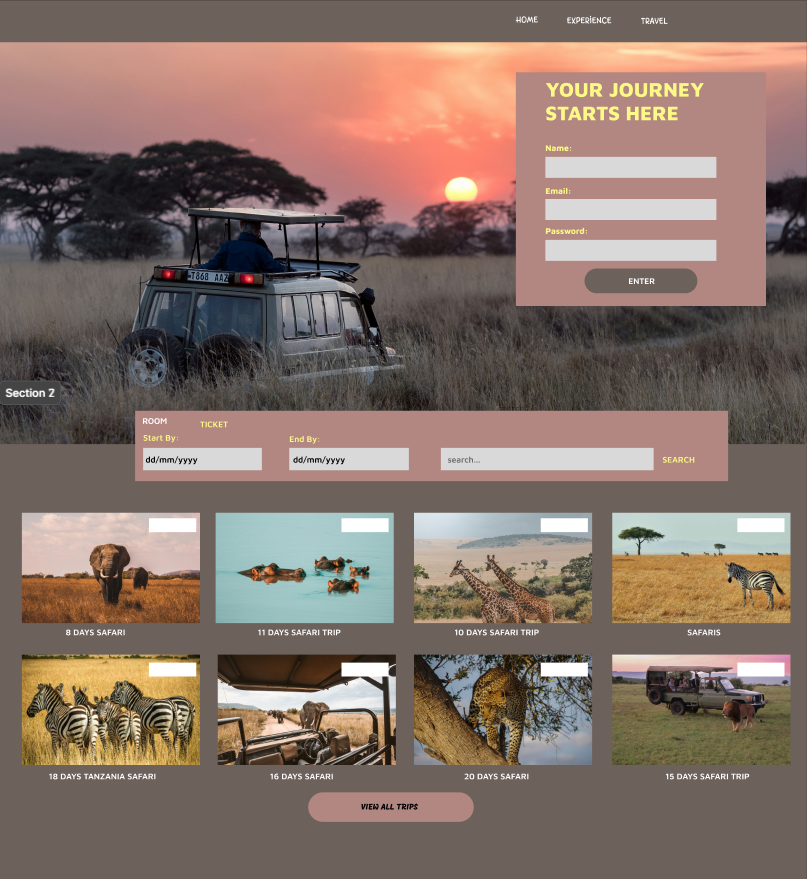
* Header: This area would contain the logo, navigation menu (Travel, Explore, Services, Experience), search bar.
* Main banner: This would contain the rotating image carousel highlighting popular destinations, with links to explore further.
* Featured sections: This section would contain the “Top destinations,” “Trending Activities” and “User recommendations” card layout.
* Footer: The footer would contain quick links, contact information, social media icons with links.

A screenshot of a website

Description automatically generated

The image above is the homepage a user sees after clicking the “EXPLORE OUR SITE” button in the landing page, the primarily focus is the two lionesses in the middle of the background which draws the user's attention. From this further text and information has been added to spark the user's imagination and being interested, specially the “Your personalized Safari Adventure. There is also a search button on the top right below the “Login and Sign Up” button which the user will find interesting and help them navigate through the website. Finally, below is the Call to Action, button stating “EXPLORE” which as the user are interested in the page there is a likelihood of them clicking this and see what is waiting for them in the next page. The homepage will serve as a gateway to the platform, highlighting animated banners, popular products, events, and deals. Overall, the design will adopt a modern aesthetic, focusing on legibility, intuitive navigation, minimalist icons, and easy accessibility. Consistent colour schemes matched to the brands identify will enhance recognition, while responsive design techniques will ensure functionality

Booking Interface:

* Booking Form: This would contain the date pickers, drop-down menus for destination and activity selection, user input fields for personal details.
* Result Section: Visually rich cards displaying activities with interactive elements for further exploration and booking directly.

Profile management Dashboard:

* User dashboard: Comprehensive view of upcoming and past trips, tailored reward programs, and personalized content.
* Personal Detail Sections: Easily editable user information fields and preferences management with a focus on user empowerment.

2.) Color Coloring Palette and Typography

Color Coloring:

* Primary color: Green
* Accent color; Blue
* Background color: Orange
* Text color: white or black

Typology:

Heading fonts: TT Espina

Body fonts: Helvetica

**DATA REQUIREMENTS**

1.) Data Enquiries:

* User Profiles:

Fields: UserID (UUID), Name, Email, PhoneNumber, Preferences (JSON), BookingHistory (Array), Feedback (Array).

* Bookings:

Fields: BookingID (UUID), UserID (Foreign Key), ActivityID (Foreign Key), Date, TotalCost, Status (Enum: Pending, Confirmed, Cancelled).

* Activities:

Fields: ActivityID (UUID), Title, Description, Location, Price, Capacity, Images (Array), Ratings (Average).

* Reviews:

Fields: ActivityID (UUID), Title, Description, Location, Price, Capacity, Images (Array), Ratings (Average).

* Reccommedations:

Fields: RecommendationID (UUID), UserID (Foreign Key), ActivityID (Foreign Key), Reason.

* Products:

Fields: product ID, seller ID, name, description, price, images, categories.

* Orders:

Fields: order ID, buyer ID, products, totalPrice, status.

Database Schema:

|  |  |
| --- | --- |
| Table Name | Description |
| Users | Stores user data such as the username, password, email, address. |
| Products | Stores product data (product ID, name, description, price, image URL). |
| Orders | Stores order data (order ID, user ID, order date, total cost, shipping address). |
| Payments | Stores payment data (payment method, transaction ID, amount). |

**ALGORITHM**

**User Authentication**

The goal is to securely authenticate users during the login process to ensure authorized access to their accounts. The process would include;

* Accept user input for email and password.
* Retrieve the user record from the database using the provided email.
* If the email or password is not found or the password is incorrect, return an error message indicating authentication failure.

def authenticate\_user(email, password):

user = find\_user\_by\_email(email)

if user and verify\_password(password, user.password):

token = generate\_jwt\_token(user.userId)

return token

return None

**Product search**

The goal is to find products based on user queries and category selections. The process would include;

* Accept user input for search query and optional category.
* Search the product database for any product.
* Collect all matching products and return them as a list to the user.
* If no products are found, return a message stating no matching products were found.

function searchProducts(query, category) {

return Product.find({

$or: [

{name: {$regex: query, $options: 'i'}},

{category: category}

]

});

}

**Shopping Cart Management**

The goal is to manage the user’s shopping cart, allowing for adding, updating, and removing products. The process would include;

* Accept the current shopping cart state and the product ID to be updated, along with the desired quantity
* Check if the product ID exists;
* Return the updated shopping cart to the user.

def update\_cart(cart, product\_id, quantity):

if product\_id in cart:

cart[product\_id] += quantity

if cart[product\_id] <= 0:

del cart[product\_id]

else:

if quantity > 0:

cart[product\_id] = quantity

return cart

Order Processing

The goal is to create an order based on items in the shopping cart and calculate the total price. The process would include;

* Accept the user ID and the shopping cart state as inputs.
* Initialize a variable to keep track of the total price (starting at zero).
* Loop through each item in the cart.
* Create a new order entry in the database.
* Save the new order and return a confirmation or the order details to the user.

function createOrder(userId, cart) {

let totalPrice = 0;

for (let productId in cart) {

const product = findProductById(productId);

totalPrice += product.price \* cart[productId];

}

const order = new Order ({ userId, products: cart, totalPrice, status: 'Pending'});

order.save();

return order;

}

Payment Processing Algorithm:

The goal is to securely process payments using a third-party payment service provider. The process include;

* Accept the order details and payment information (e.g., payment method token).
* Utilize the payment service provider’s API to create a payment transaction.
* Attempt to execute the payment transaction, handling any potential errors.
* If the payment fails, capture the error and return a message indicating that the payment was unsuccessful.

def process\_payment(order, payment\_info):

try:

stripe.Charge.create(

amount=order.totalPrice \* 100, # Amount in cents

currency='usd',

source=payment\_info['source'],

description='Order Payment'

)

order.status = 'Completed'

order.save()

return order

except stripe.error.StripeError as e:

return "Payment Failed"S

**TEST STRATEGY**

|  |  |  |  |
| --- | --- | --- | --- |
| Date of test | Components to be tested | Types of tests carried out | Dependencies |
| 17/01/2024 | Event creation form | Unit testing | Event Creation Function, Validation Function |
|  | Event creation function | Unit testing | Event Data Model, Validation Function |
|  | User registration form | Unit testing | User Registration Function, Validation Function |
|  | User registration function | Unit testing | User Data Model, Validation Function |
|  | Payment processing | Integration testing | Payment Gateway API, Event Creation Function |
|  | Ticket generation | Unit testing | Event Data Model, User Registration Function |
|  | Email Notification function | Unit testing | User Registration Function, Event Creation Function |
|  | Organizer dashboard | End-to-end testing | Event Creation Function, User Registration Function, Payment Processing |
|  | Attendee Registration | End-to-end testing | User Registration Function, Payment Processing, Ticket Generation |
|  | Payment Processing | Security testing | Payment Gateway API, Payment Data Model |
|  | User data storage | Security testing | User Data Model, Database Connection |
|  | Event data storage | Security testing | Event Data Model, Database Connection |
|  | System Performance | Performance Testing | Event Creation Function, User Registration Function, Payment Processing |
|  | System Compatibility | Compatibility testing | Different browsers and devices |
|  | System Usability | Usability testing | User interface, Navigation, Feedback |

* Unit Testing: Testing individual components or functions to ensure they work as expected.
* Integration Testing: Testing how different components or functions interact with each other.
* End-to-End Testing: Testing the entire system from start to finish to ensure it works as expected.
* Security Testing: Testing the system for vulnerabilities and security risks.
* Performance Testing: Testing the system's performance under different loads and conditions.
* Compatibility Testing: Testing the system's compatibility with different browsers and devices.
* Usability Testing: Testing the system's user interface and navigation to ensure it is user-friendly.