Hackthone 3

Day 2 Assignment: Planning the Technical Foundation for the Car Rental Web

Overview:

The Car Rental Website will allow users to browse, rent, and manage car bookings efficiently. The website should prioritize scalability, responsiveness, and ease of use.

Key Technical Requirements:

- **Frontend Framework:** React.js (with Next.js for SSR and routing)
- **Styling:** Tailwind CSS for a clean, responsive UI
- **Sanity:** Sanity for storage and API
- **Authentication:** Next auth for authentication (e.g., Google Login)
- **Payment Integration:** Stripe for secure payments
- **Deployment Platform:** Vercel for frontend
- **Version Control:** Git (GitHub for repository hosting)

Design System Architecture

Frontend:

- React components for modularity
- Context API for state management
- API calls to backend services for data retrieval

Backend:

- RESTful APIs
- Middleware for input validation

Database:

Schema design (details in Section 6)

Workflow:

1. User interaction triggers frontend logic.
2. API calls send requests to the backend.
3. Backend processes data, interacts with the database, and returns responses.
4. Frontend updates UI with the response data.

Plan API Requirements

1. **User Management:**
   - **POST** `/register`: Register a new user.
   - **POST** `/login`: Authenticate a user.
   - **GET** `/profile`: Fetch user profile details.
2. **Car Listings:**
   - **GET** `/cars`: Retrieve a list of available cars.
   - **GET** `/cars/:id`: Fetch details of a specific car.
3. **Bookings:**
   - **POST**`/bookings`: Create a new booking.
   - **GET** `/bookings`: Retrieve all bookings for a user.
   - **DELETE** `/bookings/:id`: Cancel a booking.
4. **Payments:**
   - **POST** `/payments`: Process a payment.

Setup Instructions:

1. Clone the repository from GitHub.
2. Install dependencies: `npm install`
3. Set up `.env` file

Collaborate and Refine

Team Collaboration Plan:

1. Use GitHub Issues to track bugs and features.
2. Conduct daily stand-ups to discuss progress and blockers.
3. Utilize Figma for UI/UX collaboration.

Feedback Loop:

1. Share the initial prototype with stakeholders for feedback.

2. alteratively refine features based on feedback and testing.

Schema Details

User Schema:

```
const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  phone: { type: String },
  createdAt: { type: Date, default: Date.now }
});
```

Car Schema:

```
const carSchema = new mongoose.Schema({
  name: { type: String, required: true },
  brand: { type: String, required: true },
  pricePerDay: { type: Number, required: true },
  availability: { type: Boolean, default: true },
  description: { type: String },
  images: [String],
  createdAt: { type: Date, default: Date.now }
});
```

Booking Schema:

```
const bookingSchema = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
```

```
  carId: { type: mongoose.Schema.Types.ObjectId, ref: 'Car', required: true },
  startDate: { type: Date, required: true },
  endDate: { type: Date, required: true },
  totalPrice: { type: Number, required: true },
  status: { type: String, default: 'Pending' },
  createdAt: { type: Date, default: Date.now }
});
```