Design Document for Tourist Information Application

1. Overview

The Tourist Information Application is a stateless mobile application allowing users to log in using a QR code, view their current location on a map, and access information about nearby tourist places through reading or audio options.

2. Architecture

2.1 System Components

- **Frontend**: React Native application for user interaction.
- **Backend**: Node.js server with Express for API handling.
- **Database**: MongoDB for storing user and session data.
- **External APIs**: Google Maps API for map services, Wikipedia API for information retrieval, and a Text-to-Speech service for audio playback.

3. Component Design

3.1 Frontend

- **Login Screen**:
  - QR Code Scanner: Uses device camera to scan QR codes.
  - API Call: Sends QR code data to the backend for authentication.
- **Main Map Screen**:
  - Map Display: Integrates Google Maps to show current location.
  - Pin Display: Shows pins for nearby tourist places.
  - Pin Interaction: Allows users to click pins to access information.
- **Information Screen**:
  - Read Option: Displays text summary from Wikipedia.
  - Hear Option: Provides audio controls (play, pause, stop) for Text-to-Speech.

3.2 Backend

- **Authentication Service**:
  - QR Code Validation: Validates QR code data and returns session token.
- **Location Service**:
  - Current Location: Fetches user's current location.
  - Nearby Places: Retrieves nearby tourist places.
- **Information Service**:
  - Wikipedia API Integration: Fetches place summaries from Wikipedia.
  - Text-to-Speech Integration: Converts text summaries to speech.

4. Data Flow

4.1 User Login

1. **User scans QR code** using the device camera.
2. **Frontend sends QR code data** to the backend.
3. **Backend validates QR code** and returns a session token.
4. **Frontend stores the session token** for subsequent API calls.

4.2 Map Display

1. **Frontend requests current location** using device location services.
2. **Frontend calls the backend** with the session token to get nearby places.
3. **Backend fetches nearby places** using the Google Maps API.
4. **Backend returns place data** to the frontend.
5. **Frontend displays the map** with current location and pins for nearby places.

4.3 Information Access

1. **User clicks a pin** on the map.
2. **Frontend displays options** to read or hear information.
3. **Read Option**:
    - **Frontend requests summary** from the backend.
    - **Backend fetches summary** from Wikipedia API.
    - **Backend returns summary** to the frontend.
    - **Frontend displays the summary** in a new page.
4. **Hear Option**:
    - **Frontend requests audio** from the backend.
    - **Backend fetches summary** from Wikipedia API.
    - **Backend converts summary** to speech using Text-to-Speech service.
    - **Backend returns audio URL** to the frontend.
    - **Frontend plays the audio** with controls (play, pause, stop).

5. APIs

5.1 Authentication API

- **Endpoint**: `/api/login`
- **Method**: POST
- **Request**: `{ "qrCodeData": "string" }`
- **Response**: `{ "sessionToken": "string" }`

5.2 Location API

- **Endpoint**: `/api/location`

- **Method**: GET
- **Headers**: `{ "Authorization": "Bearer sessionToken" }`
- **Response**: `{ "currentLocation": { "lat": "number", "lng": "number" }, "nearbyPlaces": [ { "name": "string", "lat": "number", "lng": "number" } ] }`

5.3 Information API

- **Endpoint**: `/api/info`
- **Method**: GET
- **Headers**: `{ "Authorization": "Bearer sessionToken" }`
- **Query Params**: `placeId`
- **Response**: `{ "summary": "string" }`

5.4 Text-to-Speech API

- **Endpoint**: `/api/speech`
- **Method**: GET
- **Headers**: `{ "Authorization": "Bearer sessionToken" }`
- **Query Params**: `text`
- **Response**: `{ "audioUrl": "string" }`

6. User Interface Design

6.1 Login Screen

- **QR Code Scanner**: Centered scanner view with a button to trigger the scan.

6.2 Main Map Screen

- **Map Display**: Full-screen map showing current location.
- **Pins**: Icons representing tourist places.
- **Pin Interaction**: Popup with options to read or hear information.

6.3 Information Screen

- **Read Option**: New page with text summary and back button.
- **Hear Option**: Audio player with play, pause, and stop buttons.

7. Security Considerations

- **Session Management**: Stateless authentication using session tokens.
- **Data Encryption**: Encrypt sensitive data in transit using HTTPS.
- **Access Control**: Ensure only authenticated users can access APIs.

8. Performance Considerations

- **Caching**: Use caching for frequently accessed data like place summaries.
- **Load Balancing**: Distribute requests across multiple servers to handle high traffic.
- **Scalability**: Design backend services to scale horizontally.

9. Testing

9.1 Unit Testing

- Test individual components and services for functionality.

9.2 Integration Testing

- Test interaction between frontend and backend components.

9.3 User Acceptance Testing

- Conduct testing with actual users to ensure the application meets requirements.

10. Deployment

10.1 Infrastructure

- Deploy frontend to a cloud-based platform like AWS Amplify or Firebase.
- Deploy backend to a cloud provider like AWS, Google Cloud, or Azure.

10.2 Monitoring

- Implement monitoring tools to track application performance and errors.