CafeQuest

Your personal coffee shop companion

Product Requirements Document

Project Type	Full-stack Mobile Application
Platform	React Native (iOS & Android)
Backend	Node.js + Express
Database	MongoDB
Timeline	15 days
Deadline	21st January

1. Problem Statement

Coffee lovers visit many cafes but often struggle to:

- Remember which cafes they've visited and what they liked
- Keep track of cafes they want to visit
- Find recommendations from others

CafeQuest is a personal cafe journal and discovery app that solves these problems.

2. Target Users

- Coffee enthusiasts and cafe hoppers
- Students who study at cafes
- Remote workers looking for work-friendly spots
- Anyone who enjoys exploring local cafes

3. Core Features

3.1 User Authentication

- Email/password signup and login
- JWT-based authentication
- User profile with avatar

3.2 Add Cafe Entry

- Cafe name and location
- Photo upload from camera/gallery
- Rating (1-5 scale)
- Tags: wifi, quiet, aesthetic, good-coffee, pet-friendly
- Personal notes/review
- Status: Visited or Want to Visit

3.3 Cafe Collection

- Grid view of saved cafes with photos
- Filter by visited/wishlist status
- Filter by tags
- Search by cafe name

3.4 Cafe Details

- Full photo view with all details
- Edit and delete functionality

3.5 Discover Feed

- Browse public cafe entries from other users

- Save cafes to personal wishlist

3.6 Profile

- View total cafes visited
- Most used tags
- Account settings

4. Tech Stack

Layer	Technology
Frontend	React Native + Expo
Navigation	React Navigation
State Management	Context API
Image Upload	Cloudinary
Backend	Node.js + Express.js
Database	MongoDB + Mongoose
Authentication	JWT + bcrypt
Hosting	Render + MongoDB Atlas

5. Database Schema

Users Collection

```
{ _id, email, password (hashed), username, avatar, createdAt }
```

Cafes Collection

```
{ _id, userId, name, location, photo, rating (1-5), tags[], notes, status
(visited/wishlist), isPublic, visitedAt, createdAt }
```

6. API Endpoints

Method	Endpoint	Description
POST	/api/auth/register	Create new user
POST	/api/auth/login	Login user
GET	/api/auth/me	Get current user
GET	/api/cafes	Get user's cafes

GET	/api/cafes/:id	Get single cafe
POST	/api/cafes	Add new cafe
PUT	/api/cafes/:id	Update cafe
DELETE	/api/cafes/:id	Delete cafe
GET	/api/discover	Get public cafes feed
POST	/api/upload	Upload image

7. App Screens

Screen	Description
Splash	App logo and loading
Login/Signup	Authentication forms
Home	Grid of saved cafes
Add Cafe	Form with image picker and rating
Cafe Details	Full view with edit/delete
Discover	Public feed from other users
Profile	User stats and settings

8. Project Timeline

Days	Task
1-2	Project setup and backend boilerplate
3-4	Authentication system
5-6	Cafe CRUD APIs and database models
7	Image upload integration
8-9	Frontend auth screens and navigation
10-11	Add cafe form and home screen
12	Cafe details screen
13	Discover feed
14	Profile page and UI improvements

15	Testing and bug fixes

9. Risks and Mitigation

Risk	Mitigation
Image upload complexity	Use Cloudinary (simple API, free tier)
Time constraints	Prioritize core features, skip Discover if needed
Database setup	Use MongoDB Atlas (cloud-hosted)

10. Success Criteria

- User can sign up and log in
- User can add a cafe with photo
- User can view, edit, and delete cafe entries
- User can filter cafes by tags and status
- Discover feed displays public entries
- App runs without crashes on iOS and Android