Software Solution Architect Mentoring Program - Assignment #1

Table of Contents

1. Project Overview

1.1. Client Need

A mobile tourist application with the following core features:

1.1.1. Primary Function

- Take a photo of an attraction (landmark, scenery, etc.)
- Receive description via voice-over and/or text
- Replaces the current flow of: photo → ChatGPT/Wikipedia → manual search

1.1.2. Secondary Function - Partnership Ecosystem

- Advertise partner businesses near attractions
- Example: Photo of Eiffel Tower → description + recommendation for partner cafeteria nearby
- Partners are **NOT** random businesses but signed partners

1.1.3. Tertiary Function - Discount System

- Show QR code at partner locations
- Receive discounts when scanned

1.2. Scope

- Global ambitions (from famous attractions like Eiffel Tower to small local vineyards in Spain)
- Some attractions may lack Wikipedia pages
- Partners will provide their own information
- Geo-location is potentially important
- All functionality should be **within the mobile app** (no external website visits)

2. Assignment Structure

2.1. Part 1: Questions (Due: Next Week)

Goal: Extract missing information through questions or assumptions

2.1.1. What's Missing

- Back-end infrastructure details
- Non-functional requirements (performance, scalability, etc.)
- How LLM integration works
- Partnership initiation and management process
- Partnership network operations
- Security guardrails and rate limiting
- What happens in edge cases (unknown attractions, spam requests, etc.)

2.1.2. Deliverable

- List of questions
- Optional: Schedule recorded Q&A session with mentor (acting as client)

2.2. Part 2: Solution Design (Due: Week After - ~2 weeks total)

2.2.1. Required Components

1. Assumptions

Document what you're assuming since you won't have 100% information

2. High-Level Component Design

- What talks to what

Key flows and use cases

- Simple scenario: User takes photo → what happens?
- Complex scenario: User in mountain village photographs partner vineyard → what happens?

3. Deep Dive on Key Areas

- Infrastructure design
- Back-end configuration
- Key integrations (especially LLM: Anthropic API? OpenAI? Self-hosted open source on EC2?)
- Rationale for technical choices

4. Cost Breakdown

- **CAPEX (Capital Expenses):** Development costs (e.g., 5 developers × hourly rate × time)
- **OPEX (Operational Expenses):** Monthly recurring costs (cloud infrastructure, licenses, etc.)

5. Development Milestones

- POC (Proof of Concept) - what features?
- MVP (Minimum Viable Product for App Store) - what features?
- Full-scale release - what features?
- Out of scope items
- Feature priorities
- High-level epic structure

6. Business Flows

Cover various scenarios

7. Additional Enhancements

You CAN propose improvements if you see gaps (e.g., security, rate limiting, etc.)

3. Format Notes

- Informal structure (not a formal RFP)
- But technically complete
- Free-form document
- Reasonable complexity (achievable by engineering team in viable timeframe)

4. Key Insights from Mentor

4.1. Question Development Approaches

1. Try to design solution with available info → identify gaps → questions emerge naturally
2. Push system boundaries → edge cases reveal questions
    - What if no attraction found?
    - What if spam?
    - What if conflicting requirements?

4.2. Important Considerations Mentioned

- Security guardrails and rate limiting (to prevent DoS and unbearable costs)
- Client may not know all answers
- Client descriptions may be contradictory
- Your job to provide reasonable solutions and flag missing critical items

5. Timeline

Timeframe	Deliverable
Next week	Questions prepared + optional Q&A session
Week after (~2 weeks total)	Solution design document

6. Next Steps

Formulate questions for Part 1 based on the identified gaps and edge cases.