Lab 1 Assignment

Yelp Prototype - Using FastAPI and ReactJS + Agentic AI

Due Date: March 24, 2026, 11:59 PM Points: 40 points

Prerequisites

- You should be able to run basic Python and React applications.
- You must be familiar with JavaScript and Python programming.

Yelp Prototype - Project Overview

Develop a Yelp-style restaurant discovery and review platform using React for the frontend and Python (FastAPI) for the backend.

The application must support two main personas: User (Reviewer) and Restaurant Owner.

Required Features

User (Reviewer) Features

1. Signup

- User signs up with name, email ID, and password
- Store passwords securely using bcrypt

2. Login/Logout

- Implement basic session-based or JWT authentication

3. Profile Page

- Display user details and upload profile picture
- Update profile information: name, email, phone number, about me, city, country, languages, gender
- Country field should be a dropdown list; state should be abbreviated

4. User Preferences

Users can set and save preferences for the AI assistant feature:

- Cuisine preferences (e.g., Italian, Chinese, Mexican, Indian, Japanese, American)
- Price range preference ($–$$$$)
- Preferred location(s) / search radius
- Dietary needs/restrictions (vegetarian, vegan, halal, gluten-free, kosher, etc.)
- Ambiance preferences (casual, fine dining, family-friendly, romantic, etc.)
- Sort preference (rating, distance, popularity, price)

5. Restaurant Search Page/Dashboard

Search available restaurants by:

- Restaurant name
- Cuisine type
- Keywords (e.g., 'quiet', 'family-friendly', 'outdoor seating', 'wifi')
- Location (city/zip)

6. Restaurant Details View
View details such as:

- Restaurant name, cuisine type, address/location, description
- Hours of operation, contact information, photos (optional)
- Average rating, review count, and list of reviews (with ratings, comments, date)

7. Add a Restaurant Listing
Users can create a new restaurant entry with:

- Restaurant name, cuisine type, address/city
- Contact info (optional), description, hours (optional), photos (optional)

8. Reviews
Users can add/edit/delete their own reviews:

- Rating (1–5 stars)
- Comment text
- Review date (server generated)
- Optionally attach photos

9. Favourites

- Mark restaurants as favorites
- Display a favourites tab

10. User History

- Display a history tab for any previous reviews or restaurants added

11. AI Assistant Chatbot

- A chatbot interface should be prominently accessible on the home screen/dashboard

Restaurant Owner Features

1. Signup

- Owner signs up with name, email ID, password, and restaurant location

2. Login/Logout

- Implement session-based or JWT authentication

3. Profile Management
View/update restaurant profile details:

- Restaurant name, cuisine type, description, location, contact info, restaurant photos, hours of operation

4. Restaurant Posting
Post restaurant for listing with details:

- Location, description, photos
- Pricing tier, amenities, cuisine type
- Contact information and hours

5. Claim/Manage Restaurant

- Owners can claim existing restaurants and manage their profiles

6. View Reviews

- Restaurant owners can view reviews for their restaurants (read-only; no deletion)

7. Owner Dashboard

- Show restaurant analytics and recent reviews

Backend Development Requirements (Python + FastAPI + MySQL)

- Python with FastAPI will be used for the backend
- Database: MySQL

Implement RESTful APIs for:

- User authentication (JWT or session-based login/signup)
- User profile management and preferences
- Restaurant posting and management
- Restaurant search and filtering
- Review creation, editing, and deletion
- Restaurant Owner dashboards
- AI Assistant chatbot endpoint

Reviews API:

- Create review (User) → linked to restaurant and user
- List reviews (for a specific restaurant)
- Update review (User - own reviews only)
- Delete review (User - own reviews only)

Security & Error Handling:

- Secure API endpoints with basic validation and error handling
- Handle errors with proper exception handling

Frontend Development Requirements (React)

Pages / Views (Minimum)

Public Pages:

- Explore/Search Page - Main landing page for discovering restaurants with search and filter functionality; display restaurant cards with key information
- Restaurant Details Page - Comprehensive view of individual restaurant; display reviews from all users; show ratings, photos, and restaurant information

User Pages (Logged In):

- Signup/Login Pages - User authentication interface with form validation and error handling
- Profile + Preferences Editor - User profile management; preferences configuration for AI assistant; update personal information and dining preferences
- Add Restaurant Form - Interface for creating new restaurant listings; upload photos and enter restaurant details
- Write Review Form - Create and submit restaurant reviews with rating system and comment section
- AI Assistant Interface - Chat-like conversational interface prominently displayed on home screen, or 'Ask Assistant' panel integrated on Explore page; real-time interaction with AI chatbot.

Owner Pages (Logged In - Optional/Stretch):

- Signup/Login Pages - Owner authentication interface with form validation and error handling
- Restaurant Profile Management - View and update restaurant details; manage restaurant information, photos, hours, and contact details
- Add/Edit Restaurant Form - Interface for posting new restaurants or editing existing listings; upload photos, set pricing tiers, and manage amenities
- Claim Restaurant Feature - Interface for owners to claim existing restaurant listings and take ownership
- Reviews Dashboard - View all reviews for owned restaurants; read-only interface with filtering and sorting capabilities
- Owner Analytics Dashboard - Display restaurant performance metrics including total views, ratings distribution, recent reviews, and overall public sentiment analysis

Frontend Standards

Responsive Design:

- Clean, modern, and responsive user interface
- Optimized for mobile, tablet, and desktop viewports
- Utilize CSS frameworks like Bootstrap or TailwindCSS

API Integration:

- Use Axios or Fetch API for all backend communications
- Implement proper error handling and loading states
- Manage asynchronous data operations efficiently

Code Architecture:

- Clear separation of concerns (components/pages/services)
- Reusable component design
- Organized folder structure for maintainability

AI Assistant Chatbot Features (Python FastAPI + Langchain + MySQL)

The AI Assistant is a conversational chatbot that helps users discover restaurants based on their preferences and natural language queries. The chatbot should be prominently displayed on the home screen/dashboard.

Core Functionality

- Fetches the user's saved preferences from the database (cuisine preferences, price range, dietary needs, location, ambiance preferences)
- Interprets natural language queries from users
- Searches the restaurant database based on interpreted filters and user preferences
- Ranks and recommends restaurants using both the query context and saved preferences
- Provides personalized recommendations with reasoning
- Supports multi-turn conversations to refine recommendations

Example Interactions

Example 1:
User Query: 'I'm looking for a place for dinner tonight'
AI Response: Based on your preferences for Italian cuisine and mid-range pricing, I recommend: 1. Pasta Paradise (4.5★, $$) - 'Matches your Italian preference and budget' 2. Trattoria Roma (4.7★, $$) - 'Highly rated, Italian, within your price range'

Example 2:
User Query: 'Something romantic for an anniversary'
AI Response: For a romantic dinner, considering your preferences: 1. Candlelight Bistro (4.8★, $$$) - 'Romantic ambiance, French cuisine, highly rated' 2. Sunset Terrace (4.6★, $$$) - 'Outdoor seating with views, perfect for special occasions'

Example 3:
User Query: 'I'm vegan and want something casual'

AI Response: Here are casual vegan-friendly options: 1. Green Leaf Café (4.4★, $) - '100% vegan menu, casual atmosphere' 2. Veggie Delight (4.5★, $$) - 'Extensive vegan options, relaxed setting'

Implementation Requirements

Endpoint:

- POST /ai-assistant/chat
- Input: { "message": "user query", "conversation_history": [...] }
- Output: Structured JSON response with recommendations

The AI assistant must:

- Load user preferences from the database on first query
- Use Langchain for natural language understanding and query interpretation
- Extract key information: cuisine type, price range, dietary restrictions, occasion, ambiance
- Query the restaurant database with filters
- Rank results based on relevance to query + user preferences
- Provide conversational, helpful responses
- Handle follow-up questions and refinements
- Use Tavily web search (https://www.tavily.com/#features) for additional context like current restaurant hours, special events, or trending restaurants

User Interface for AI Chatbot

The AI Assistant chatbot should be prominently displayed on the home screen/dashboard.

The chatbot interface should include:

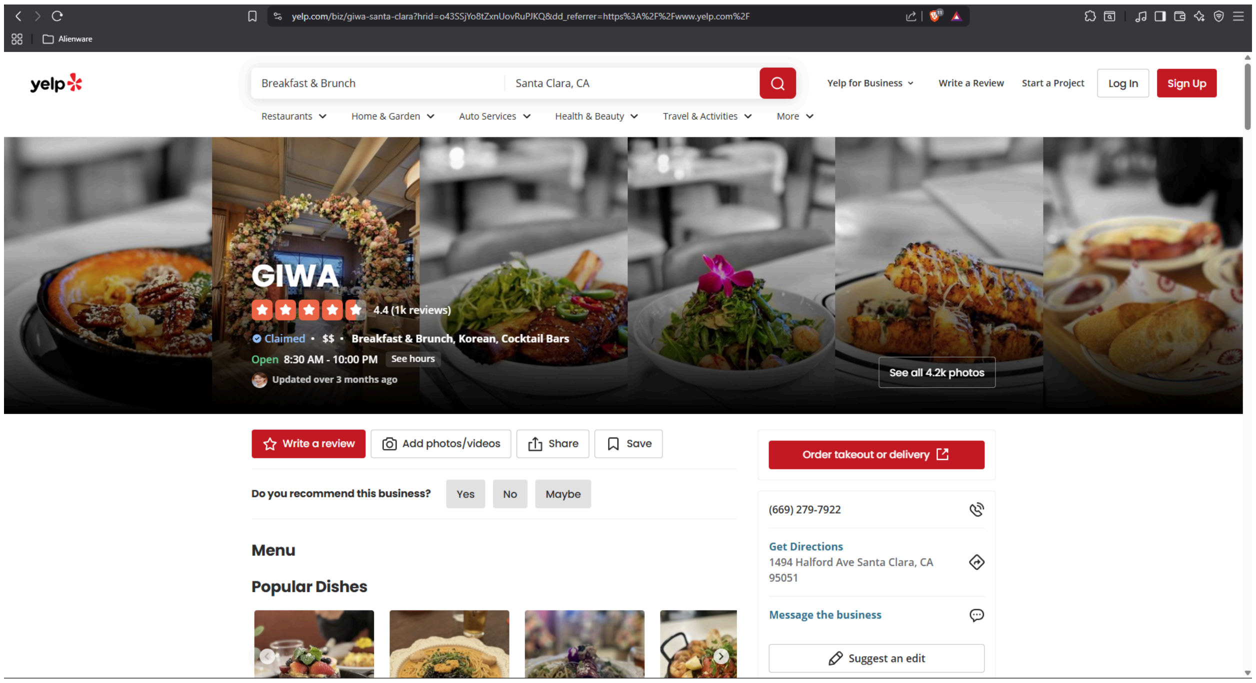
- A chat window showing conversation history
- An input field for user queries
- Display of recommended restaurants with key details (name, rating, price, cuisine)
- Clickable restaurant cards that link to full restaurant details page
- Clear indication when the AI is 'thinking' or processing
- Ability to start a new conversation/clear chat
- The chatbot should feel conversational and helpful, not robotic

Optional Enhancements:

- Add quick action buttons like 'Find dinner tonight', 'Best rated near me', 'Vegan options'

Example Reference:

visit : https://www.yelp.com/ to see the structure



API Documentation

You are required to document your API endpoints using either Swagger or Postman.

Option 1 - Swagger:

- Use Swagger to generate API documentation
- The Swagger UI should allow for testing API routes

Option 2 - Postman Collection:

- Export a Postman collection with descriptions for each endpoint
- Include request parameters, headers, and sample responses
- Submit the Postman collection along with your project

Non-Functional Requirements

- **Responsiveness:** Application must work on mobile, tablet, and desktop
- **Accessibility:** Implement semantic HTML, alt text, and keyboard navigation support
- **Scalability:** Optimize queries, avoid unnecessary data loading, ensure efficient API response times

Git Repository Guidelines

- **Commit History:** Add detailed commit messages describing changes

- **Dependencies:** Do not submit venv or __pycache__. Instead, include requirements.txt
- **README.md:** Instructions to run the application
- **Private Repository:** Invite Devdatta1999 and Saurabh2504

Report Guidelines

Submit a brief report including:

- **Introduction:** Purpose and goals of the system
- **System Design:** Explain architecture (Python, FastAPI, MySQL, React, AI Assistant Service)
- **AI Implementation:** Explain how the chatbot works and how it interprets queries and uses preferences
- **Results:** Screenshots of key screens (home page with chatbot, restaurant search, restaurant details, profile/preferences, reviews, chatbot conversation examples) and API test results

Submission Guidelines

- Upload your project report (YourName_Lab1_Report.doc) on Canvas before the deadline