

Midterm Activity – Social Coding Midterm Project

# Social Coding Selection

Select a social coding project application for your team from the below options:

* Option 1: Feature enhancements of the Lab 4.9.2 code by adding user-friendly features to the MapQuest REST API [Level of difficulty: +++]

* Option 2: Adapting the Lab 4.9.2 python framework to integrate GPT-3/GPT-4 REST API [Level of difficulty: ++++]

What were the reasons your team selected this option?

We see an excellent opportunity in this project to enhance our programming skills. Working with the MapQuest REST API and improving the existing code from Lab 4.9.2 will allow us to delve deeper into the principles of development and interaction with external web services. Through practical experience, we will gain insights into handling requests, responses, authentication, and data processing. Our analysis revealed that MapQuest REST API is related to geographical data, and we will learn how to work with coordinates, routes, geocoding, and maps. None of our team members have prior practical experience with these technologies, so we decided that it would be an interesting challenge for everyone. Working with APIs is a crucial aspect of development, and our main goal was to gain a deeper understanding of REST API principles and apply them in practice.

Describe your team's project application and its deliverables. What are the specific objectives of this application?

The project application developed by our team is a backend system designed to provide geocoding and routing services. Here are the key deliverables and objectives of the application:

1) Deliverables:

- A modular backend system that can interface with the MapQuest API to convert place names into geographical coordinates and calculate routes between locations.

- API endpoints created using Flask, which allow users to send geocoding and routing requests and receive responses in JSON format.

- Comprehensive documentation and comments within the code to ensure clarity and maintainability.

2) Objectives:

- To offer reliable geocoding services that accurately translate place names into latitude and longitude coordinates.

- To provide efficient routing solutions that can calculate the best routes based on various parameters like transportation mode and distance.

- To ensure scalability and performance so that the system can handle a growing number of requests without compromising speed or accuracy.

- To maintain high security and robust error handling to protect user data and provide a stable service.

The application is structured to be maintainable and extensible, allowing for future enhancements such as adding new features or integrating additional services. The focus on user feedback and continuous improvement ensures that the application will evolve to meet users' needs more effectively over time. The backend system is the core of the application, and while it currently does not include a frontend interface, the structure allows for easy integration with one in the future.

Overall, the application aims to provide a solid foundation for any service that requires geocoding and routing capabilities, with a focus on reliability, user experience, and adaptability to changing requirements.

Record your team member roles and skillsets

|  |  |
| --- | --- |
| Team member | Role/Knowledge/Skillset |
| Azizbek | Team lead. Back-end developer (Flask framework). Python proficiency. Web design |
| Asilbek | Front-end developer (JS, HTML, CSS). API integration |
| Birzhan | Back-end developer (Python). Python proficiency. Documentation part |
| Luca | Back-end developer (Python). MapQuest API documentation. |
|  |  |

# Strategy/Project Plan

Provide a brief description of your team’s strategy for completing this project.

Our team has developed a strategy following the Lean methodology, with a focus on MapQuest REST API. Here’s a detailed description of our plan:.

1. **Project Overview**:  
    - Define the project scope, objectives, and deliverables.

* Identify key stakeholders and their requirements.

1. **Lean Approach**:  
    - Streamline processes: Eliminate unnecessary steps and bureaucracy.

* Prioritize essential tasks: Focus on high-impact activities.
* Optimize resource utilization: Minimize waste.

1. **API Selection**:  
    - Evaluate MapQuest REST API:

Understand its capabilities, endpoints, and data formats.

Assess its suitability for our project requirements.

1. **Requirements Gathering**:  
    - Collaborate with stakeholders to collect detailed requirements.

* Define user stories and acceptance criteria.

1. **Architecture Design**:

* Plan the system architecture:

Front-end (UI/UX design, user flows).

Back-end (API integration, data storage).

* Choose appropriate technologies (e.g., Flask, JS).

1. **Development Phases**:

* Iterative development:

Build core features incrementally.

Frequent feedback loops with stakeholders.

* Implement MapQuest API integration:

Geocoding, routing, mapping, etc.

1. **Testing and Quality Assurance**:

* Conduct unit testing, integration testing, and user acceptance testing.
* Ensure API responses meet expectations.
* Address any issues promptly.

1. **Deployment and Monitoring**:

* Deploy the application to a staging environment.
* Monitor performance, security, and scalability.
* Prepare for production deployment.

1. **User Training and Documentation**:

* Create user guides and documentation.
* Train users on how to use the application.

1. **Launch and Post-Launch Activities**:

* Deploy to production.
* Monitor user feedback and address any issues.
* Continuously improve based on user needs.

# Using GitHub for Collaboration

What is the link to your GitHub repository?

<https://github.com/Gelbton/HotTeaColdBeer>

Describe how GitHub was used to:

1. Create branches (in the context of this project)
2. **project\_frontend**: This branch contains all the front-end materials.
3. **flask\_integration**: Here, we handle all the Flask-related work.
4. **method\_refactoring**: This branch focuses on describing the work related to API integration.
5. **design**: All design-related materials are stored in this branch.
6. **documentation-part**: In this branch, you’ll find all the documentation, explanations, and additional materials.

Add team members (and their branches/commits)

Project\_frontend = Asilbek

Flask\_integration = Azizbek

Documentation\_and \_design = Birzhan

Method\_refactoring = Luca

Mention pull requests, code review, merge, etc. (in the context of this project)

Pull Requests: Each team member has created branches for their respective tasks.

Code Review: Asilbek can review Azizbek's Flask integration, while Azizbek can review Birzhan's documentation and design. This helps us ensure the quality and consistency of the code throughout the project.

Continuous Integration (CI): Set up a continuous integration pipeline to automatically run tests every time a new pull request is created or updated. This helps us identify issues at an early stage and ensures that the main branch is always in a deployable state.

Documentation: I, Birzhan, who am responsible for documentation and design, continuously update the project documentation in accordance with the changes that are merged. This includes updating any design documents and the project's README.

Refactoring: Luca, who is responsible for method refactoring, regularly submits pull requests with code refactoring to improve the maintainability and performance of the codebase.