IE437 Final Project Guidelines

Solving Real-world Decision-Making Problems with AI Agent

Dept. of Industrial & Systems engineering, KAIST
Jinkyoo Park

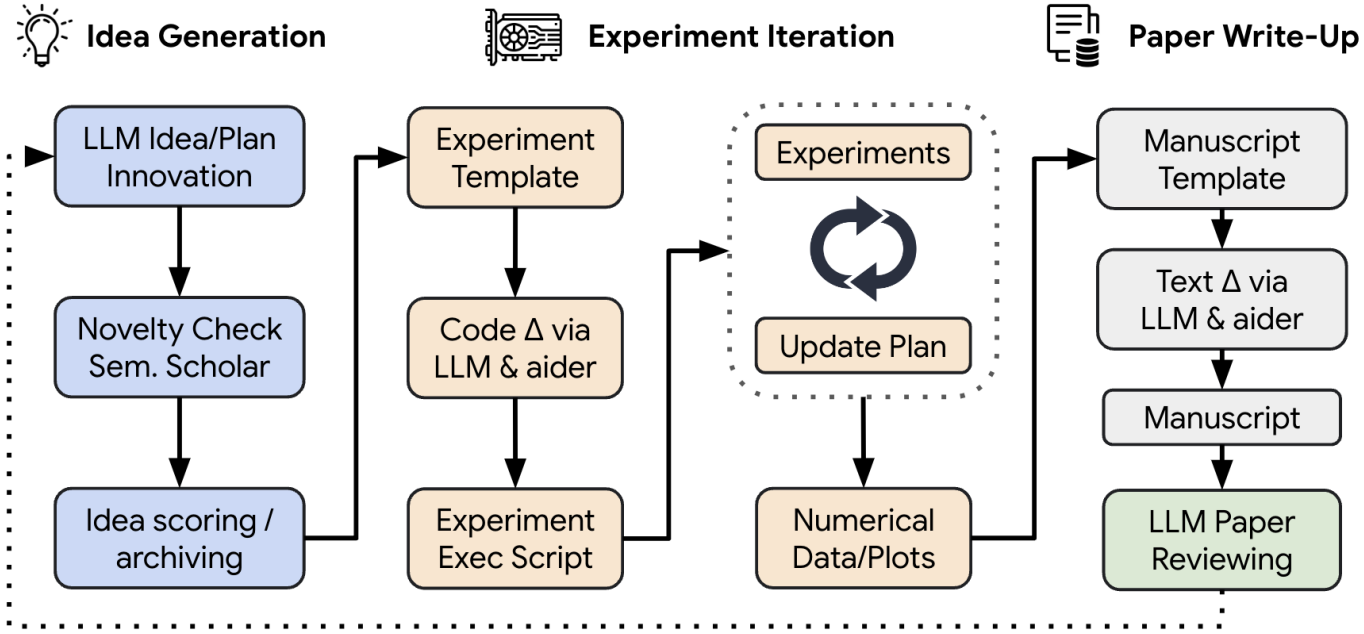
Contents

1. **Motivation of the project**
2. **Requirements**
3. **Demo**
4. **Evaluation Criteria**

Motivation

Why an AI-Agent Final Project?

- AI agents are becoming a core paradigm for real-world decision making. They can plan, reason, and interact with tools in dynamic environments.
- Traditional optimization works well, but human experts should explicitly define objectives and constraints, choose appropriate algorithms, compare the results by their own.
- LLM-based agents can adapt, interpret instructions, and improve decision quality. This project lets you explore how agents solve practical decision-making problems.



Requirements

Installation)

- We need to setup the following things:
  - **uv: https://pydevtools.com/handbook/how-to/how-to-install-uv/**
  - **git: https://git-scm.com/install/windows**

How to install uv

macOS/Linux Windows

Open PowerShell and run:

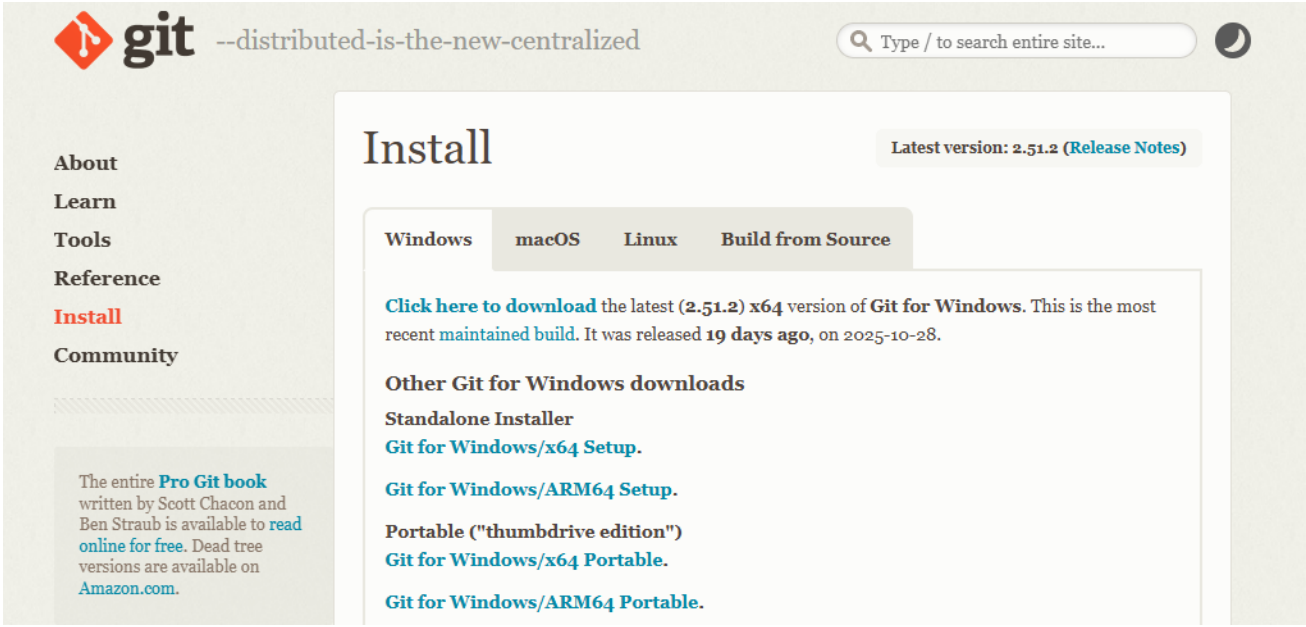
```
powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"
```

The `-ExecutionPolicy ByPass` flag allows running the installation script from the internet.

Verifying Installation

After installation, restart your terminal and verify uv is working:

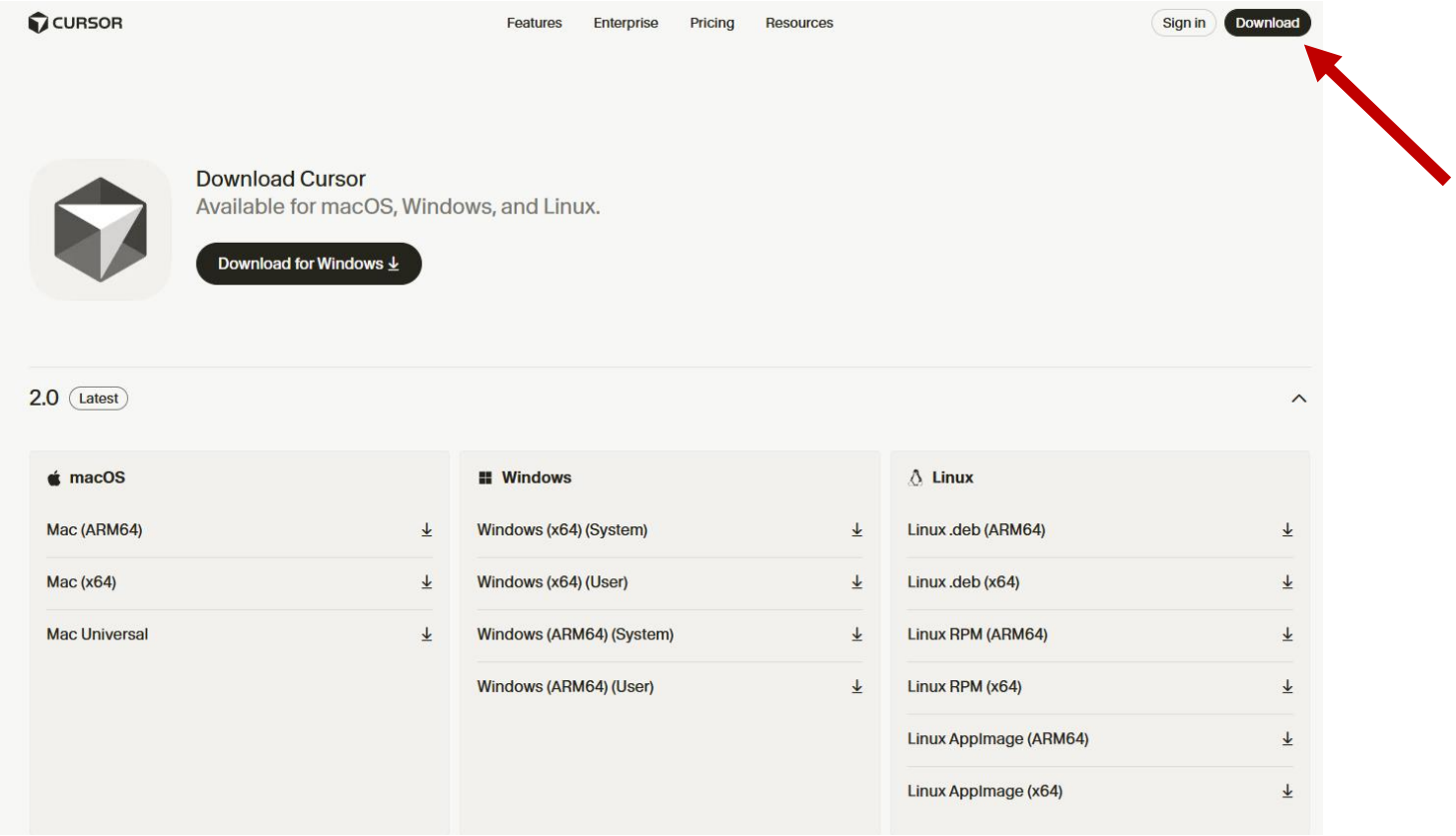
```
uv --version
```



Requirements

Installation)

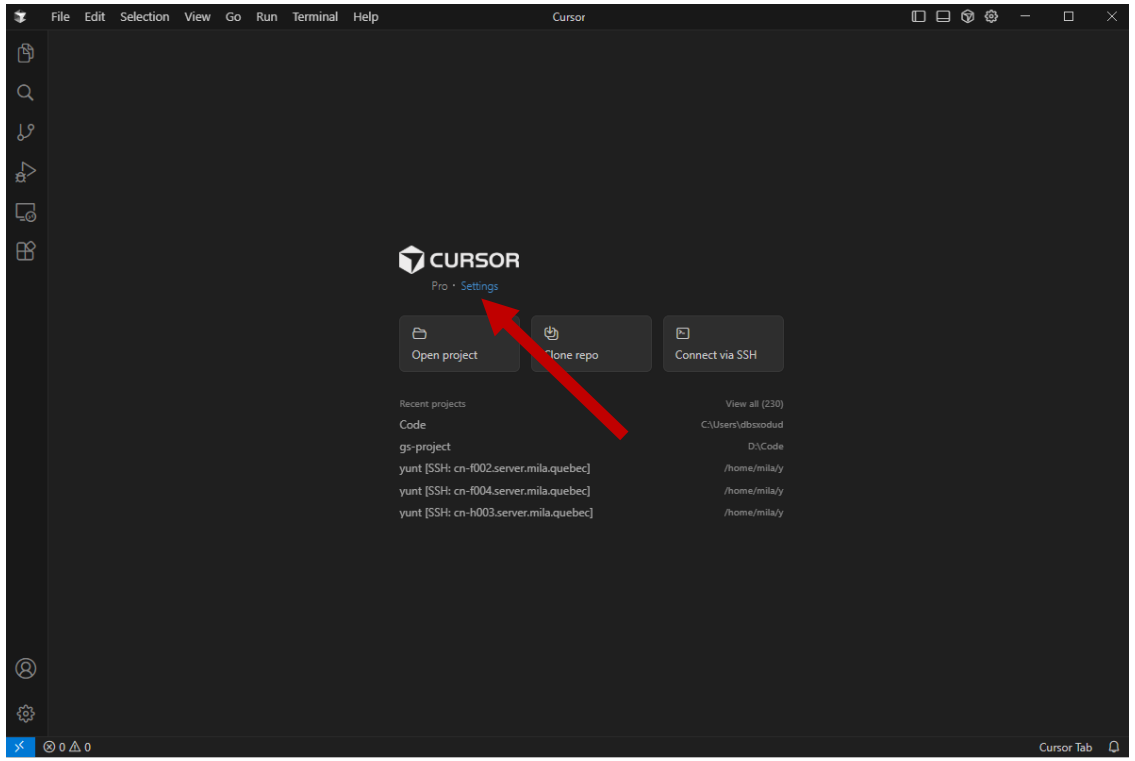
- We need to setup the following things:
  - **MCP server**
    - Step 1: Install cursor (https://cursor.com/download)

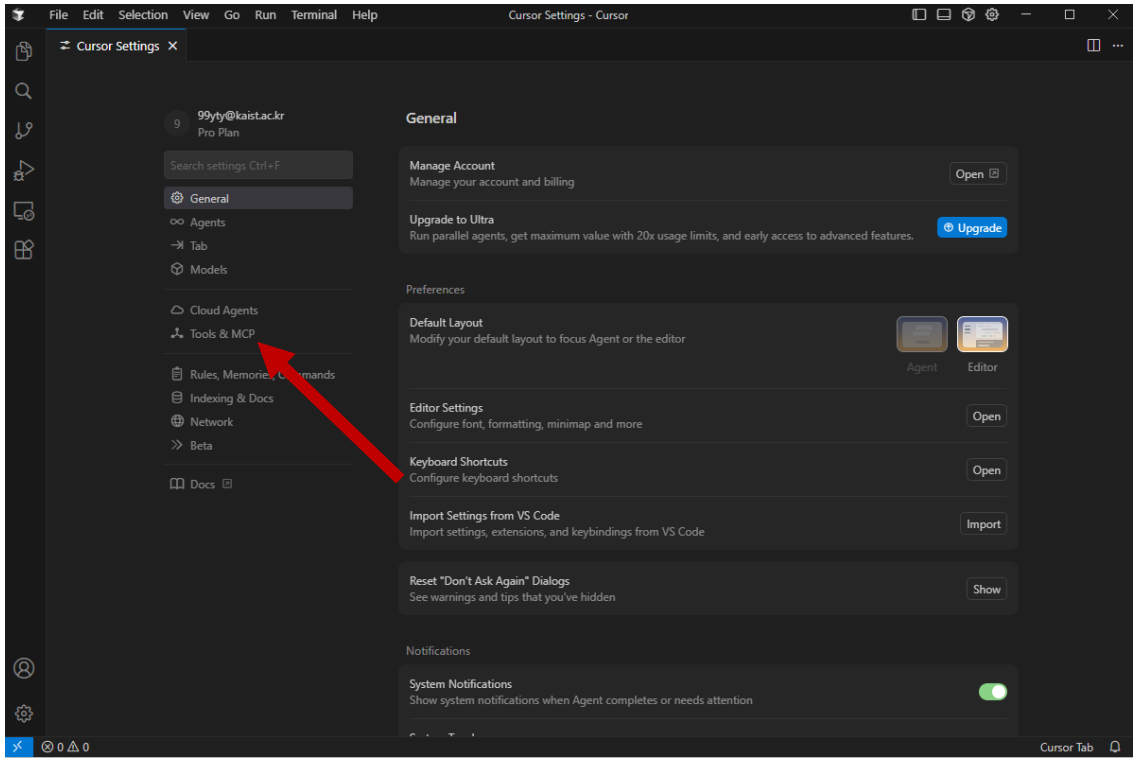


Requirements

Installation)

- We need to setup the following things:
  - **MCP server**
    - Step 2: Open cursor → Settings → Tools & MCP

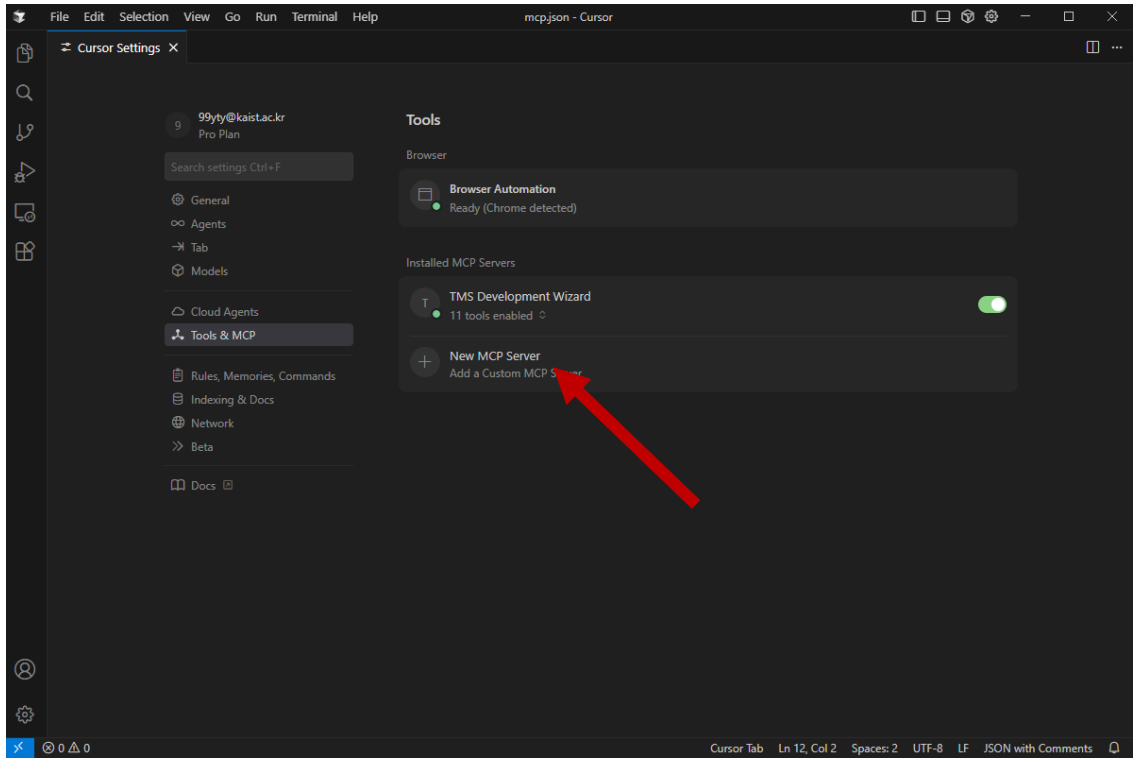


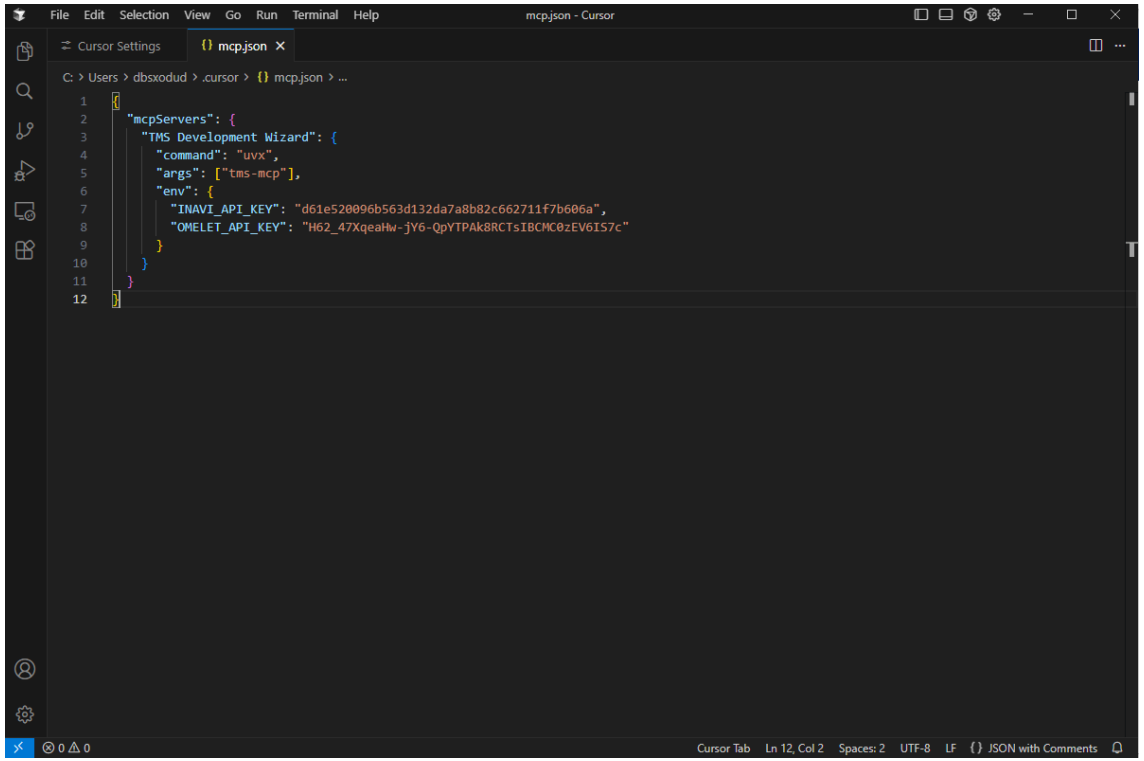


Requirements

Installation)

- We need to setup the following things:
  - **MCP server**
    - Step 3: New MCP server → copy *mcp.json, .env* provided in the zip file

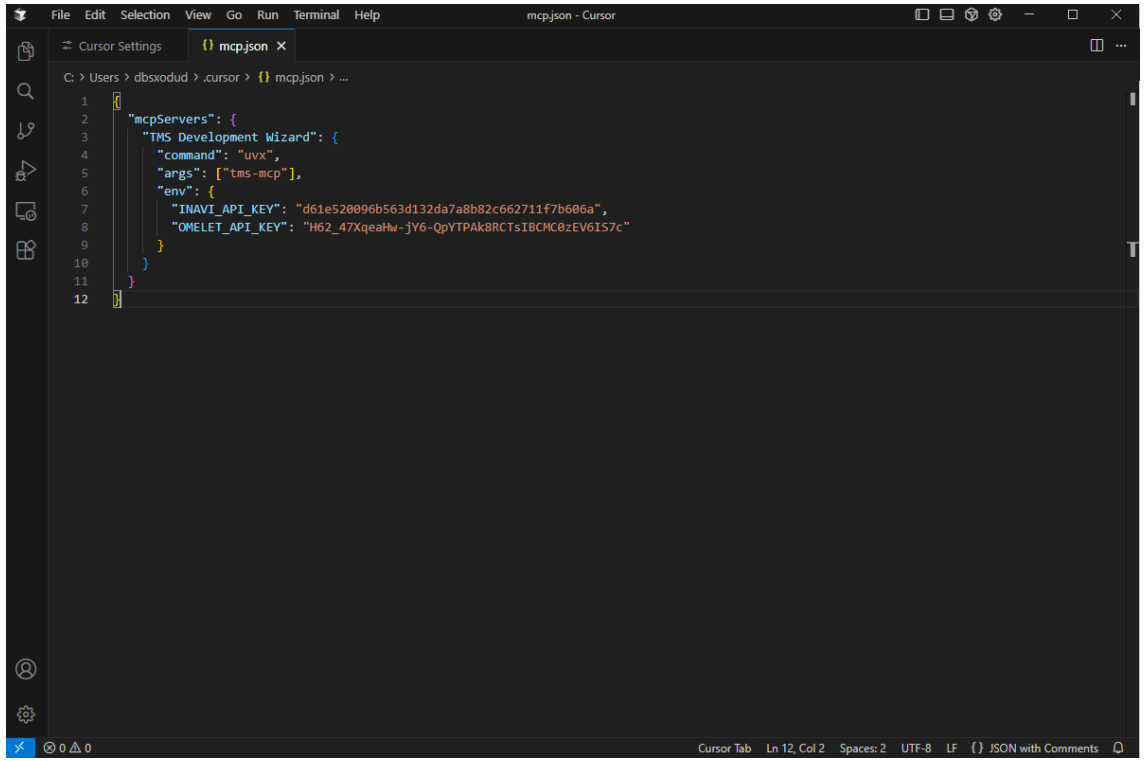




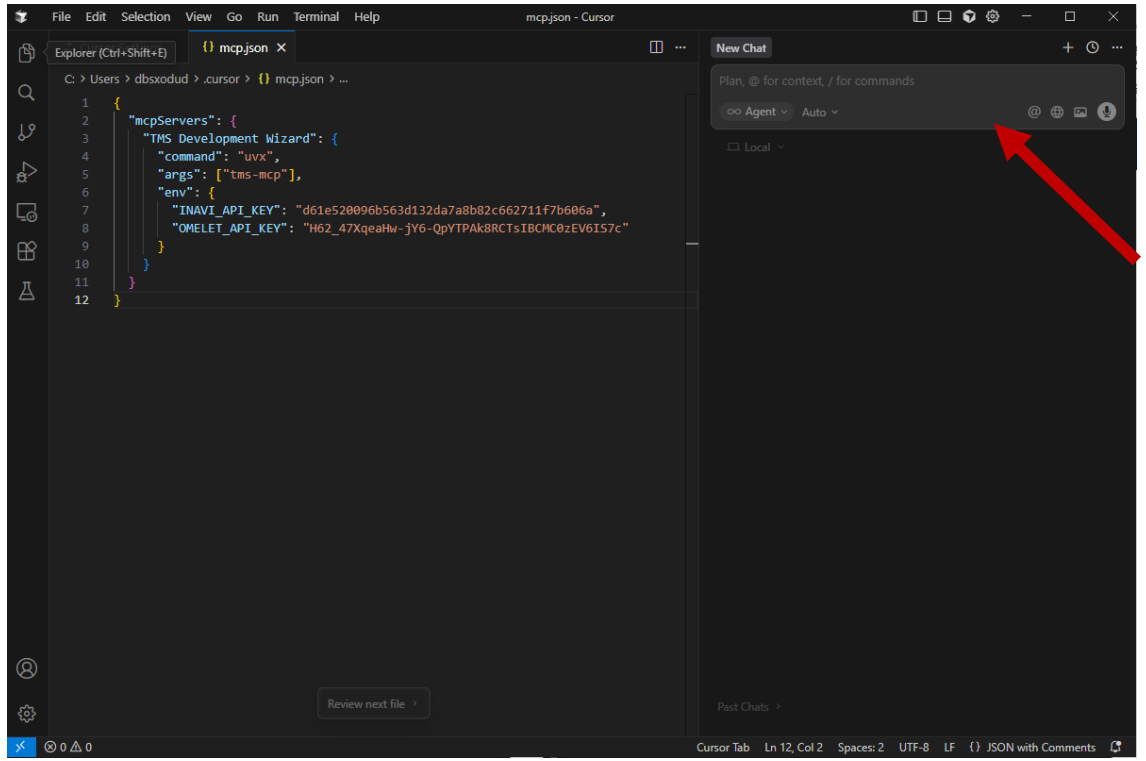
Requirements

Installation)

- We need to setup the following things:
  - **MCP server**
    - Step 4: Using Ctrl + L to start chatting with AI Agent



Ctrl + L



Demo

Tips for writing effective prompts)

- To get high-quality results from the Routing MCP Server, follow these key principles:
  - Step 1: Exploration
    - Move gradually from high-level understanding to implementation (e.g., review the API overview, check the schemas, study examples, …)
  - Step 2: Test loop
    - Adopt an iterative development cycle: Implement → Test → Fail → Debug → Retest.
    - This loop dramatically improves reliability and solution quality.
  - Step 3: Plan Before You Execute
    - For complex tasks, always begin by sending a “planning prompt”. Let the agent outline the steps first, then proceed to execution.
  - Step 4: Provide Concrete Requirements
    - Be specific about what you want. Clear constraints leads to better results.
  - Step 5: Consider stochasticity of LLM
    - LLMs are probabilistic. The same prompt may produce different outputs, so always test multiple times when needed.

Demo

Example: Seoul Delivery VRP Demo)

- Suppose that we want to make react-based dashboard with following attributes:
  - Generate N random delivery points in Seoul
  - Let use add multiple logistics centers (DCs) by clicking on the map
  - Assign each delivery point to the nearest DC and color points and DCs by assignment
  - Draw optimal routes for delivery

Problem 9 (20 Points)

Story. A convenience-store company plans to redesign its logistics system. It must decide where to locate new regional warehouses that will supply hundreds of stores across the country. This problem involves two levels of decision-making:

- **Upper level (strategic):** Select warehouse locations.
- **Lower level (operational):** For each set of chosen warehouses, determine delivery routes to stores and estimate the resulting logistics cost.

The company's goal is to minimize the overall logistics cost while maintaining timely deliveries. However, evaluating each possible location combination requires running a complex routing simulation, which is time-consuming and uncertain. You are asked to reason through this hierarchical decision problem and propose an effective strategy for solving it.

Demo

Example: Seoul Delivery VRP Demo)

- Here is an example prompt we can use:

  Below are the requirements. Please design and implement the full project based on them. I want to build a demo application with the following features: "Generate N delivery locations in Seoul → Specify logistics center locations → Assign delivery locations to centers based on road distance → Color mapping → 3-vehicle VRP optimization per center → Map visualization."

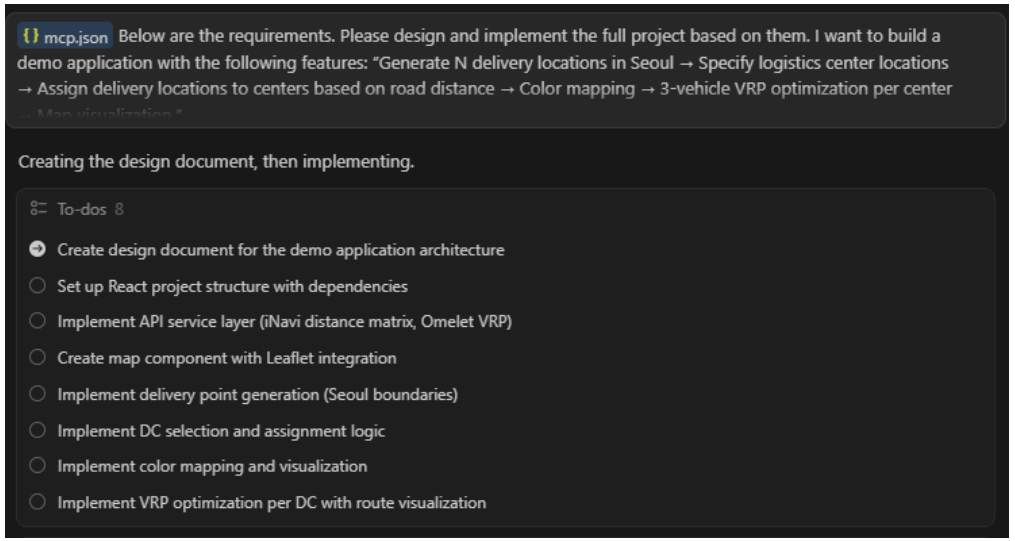
  Requirements:

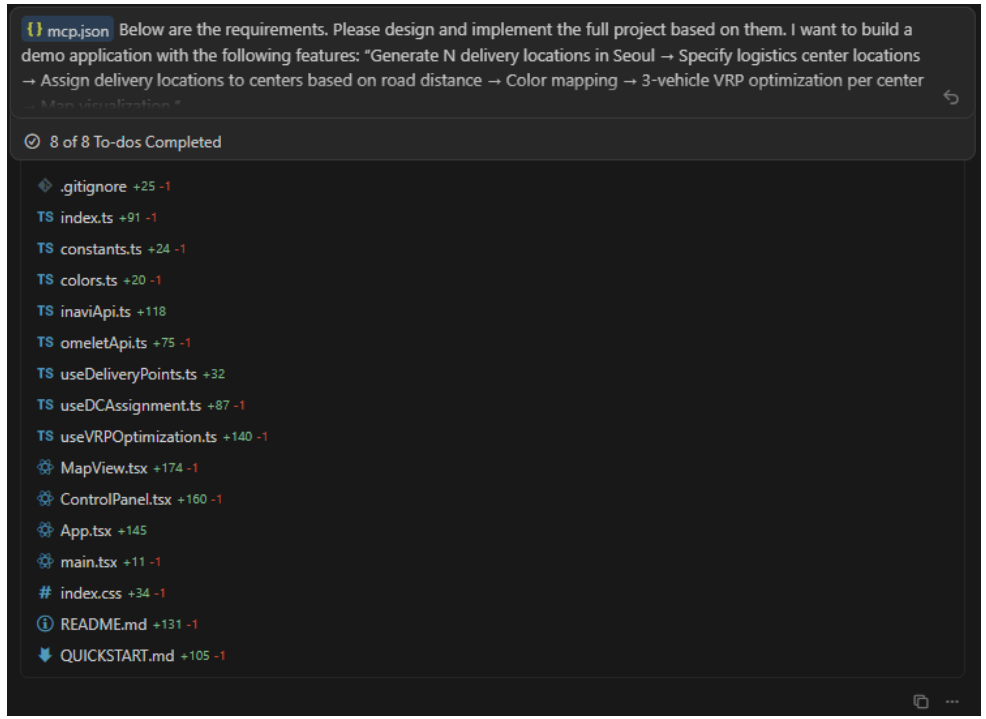
  - When the use click Number of delivery points and write a number N, generate N random delivery points (latitude/longitude) within the boundary of Seoul and display them on a map.
  - When the user clicks on the map, add that location as a logistics center (DC). Multiple DCs should be supported.
  - For every delivery point, use the iNavi road-distance API to compute the actual road travel distance to each DC, and assign the point to the nearest DC based on road distance.
  - Use color mapping so that all delivery points assigned to a DC share the same color as that DC's icon.
  - For each DC, assign 3 vehicles and call the Omelet Routing Engine /vrp API to compute optimal routes.
  - Display the resulting routes on the map as polylines. The UI should be a React-based map dashboard.
  - After the design is complete and approved, implement the application according to the design.

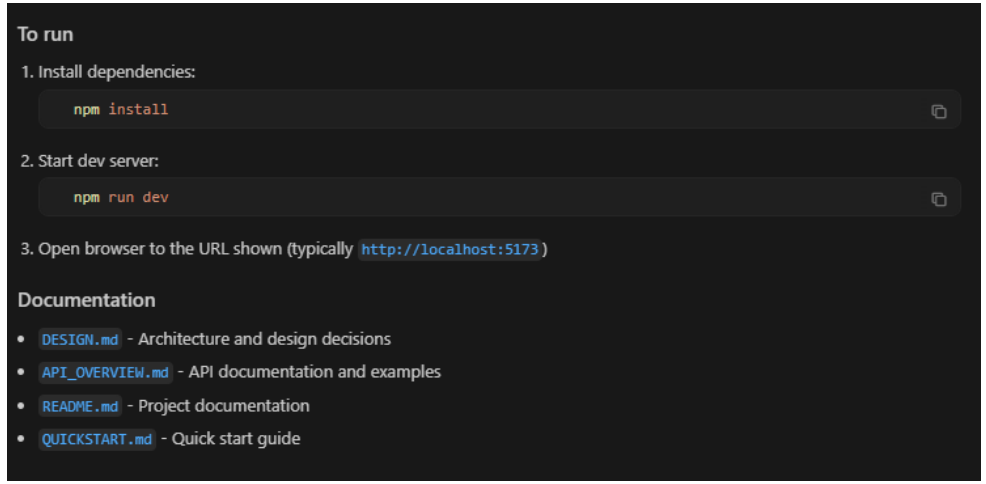
Demo

Example: Seoul Delivery VRP Demo)

- Generating….



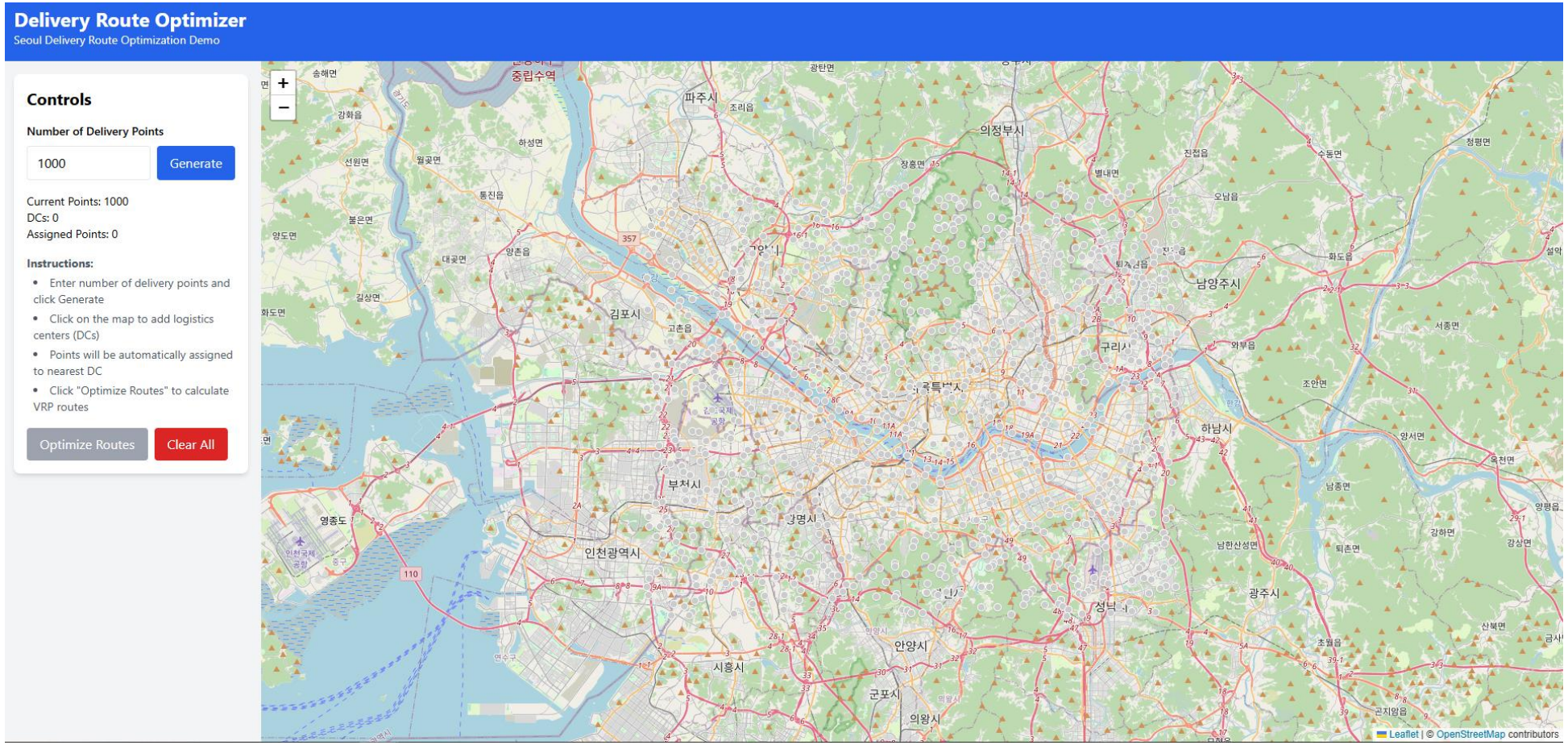




Demo

Example: Seoul Delivery VRP Demo)

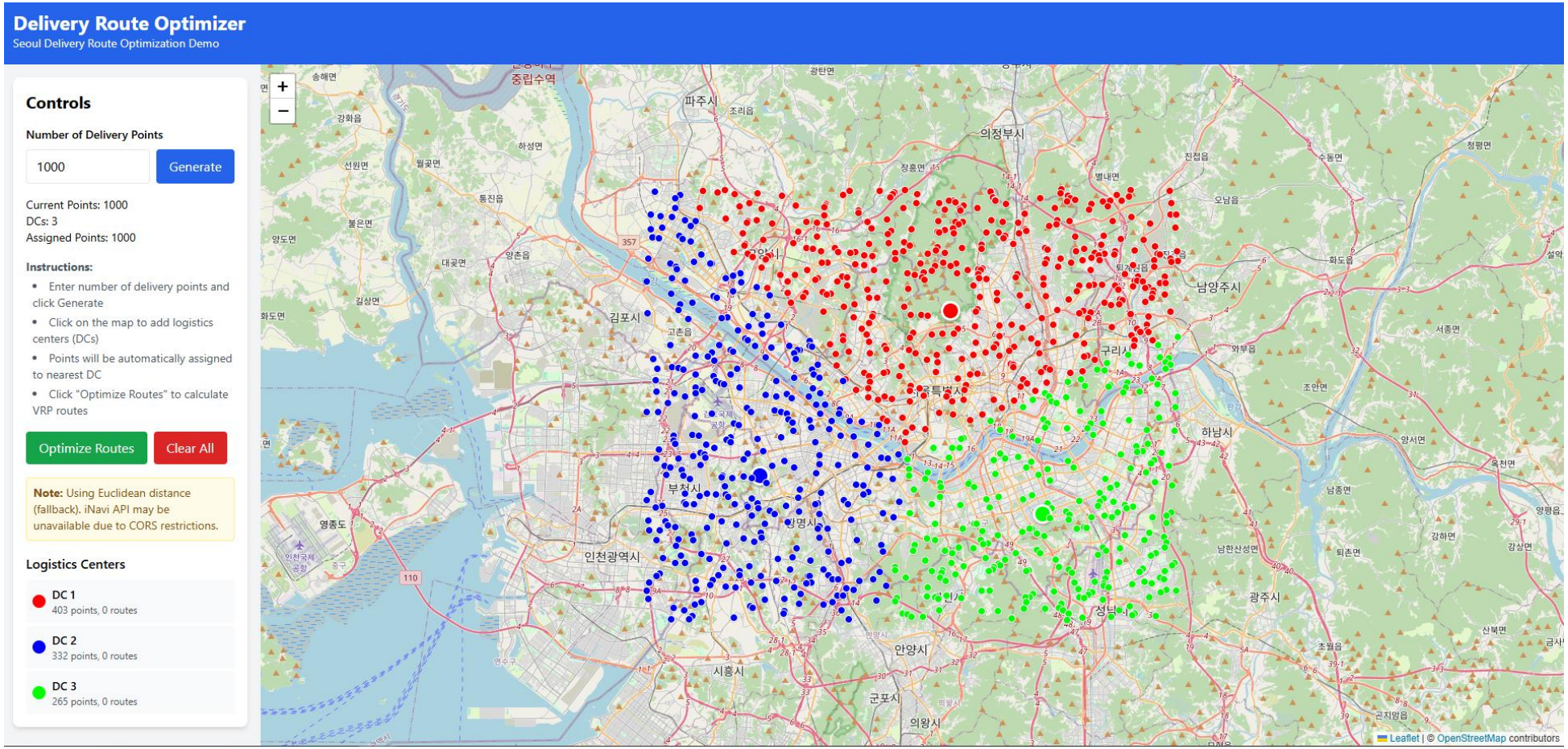
- Demo Browser



Demo

Example: Seoul Delivery VRP Demo)

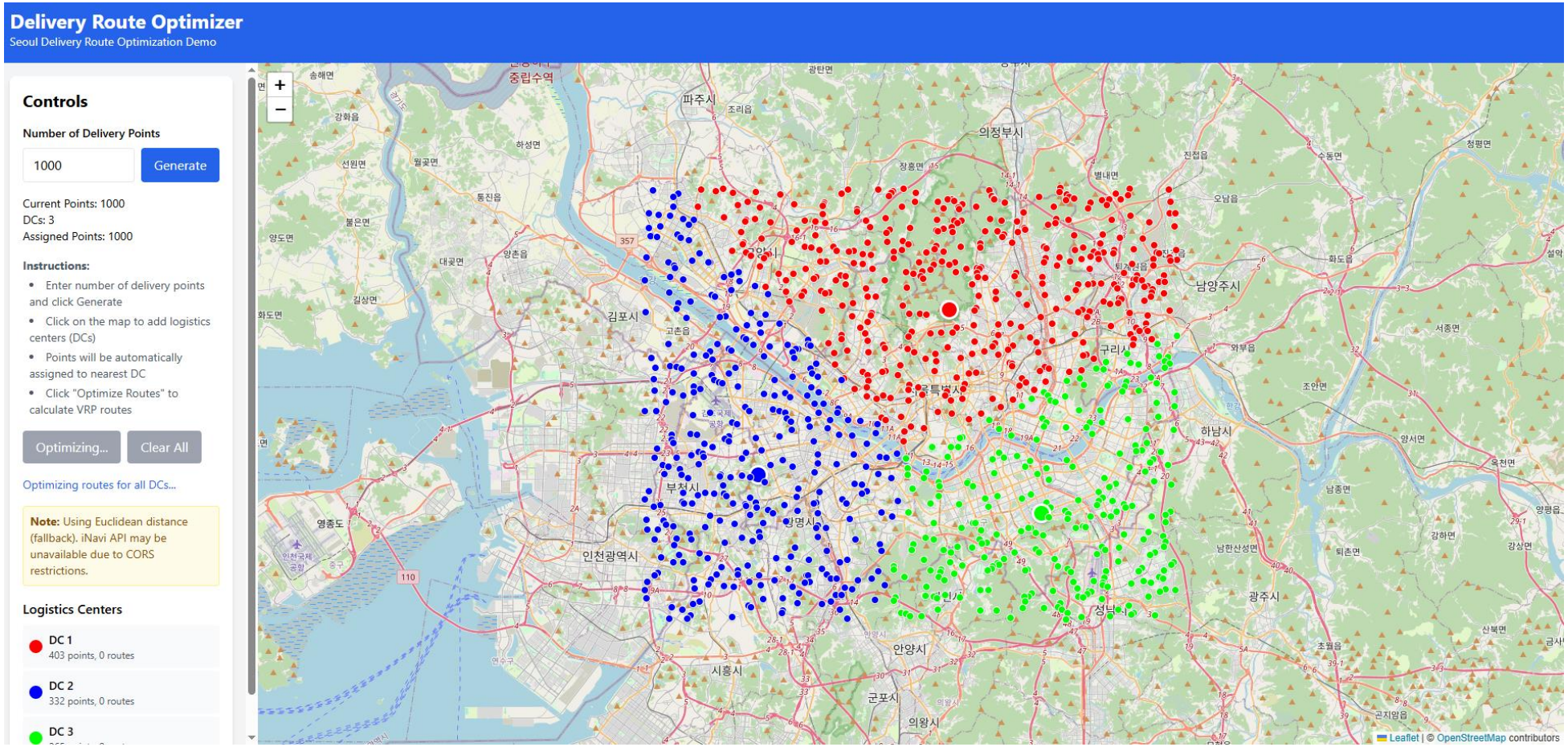
- Demo Browser



Demo

Example: Seoul Delivery VRP Demo)

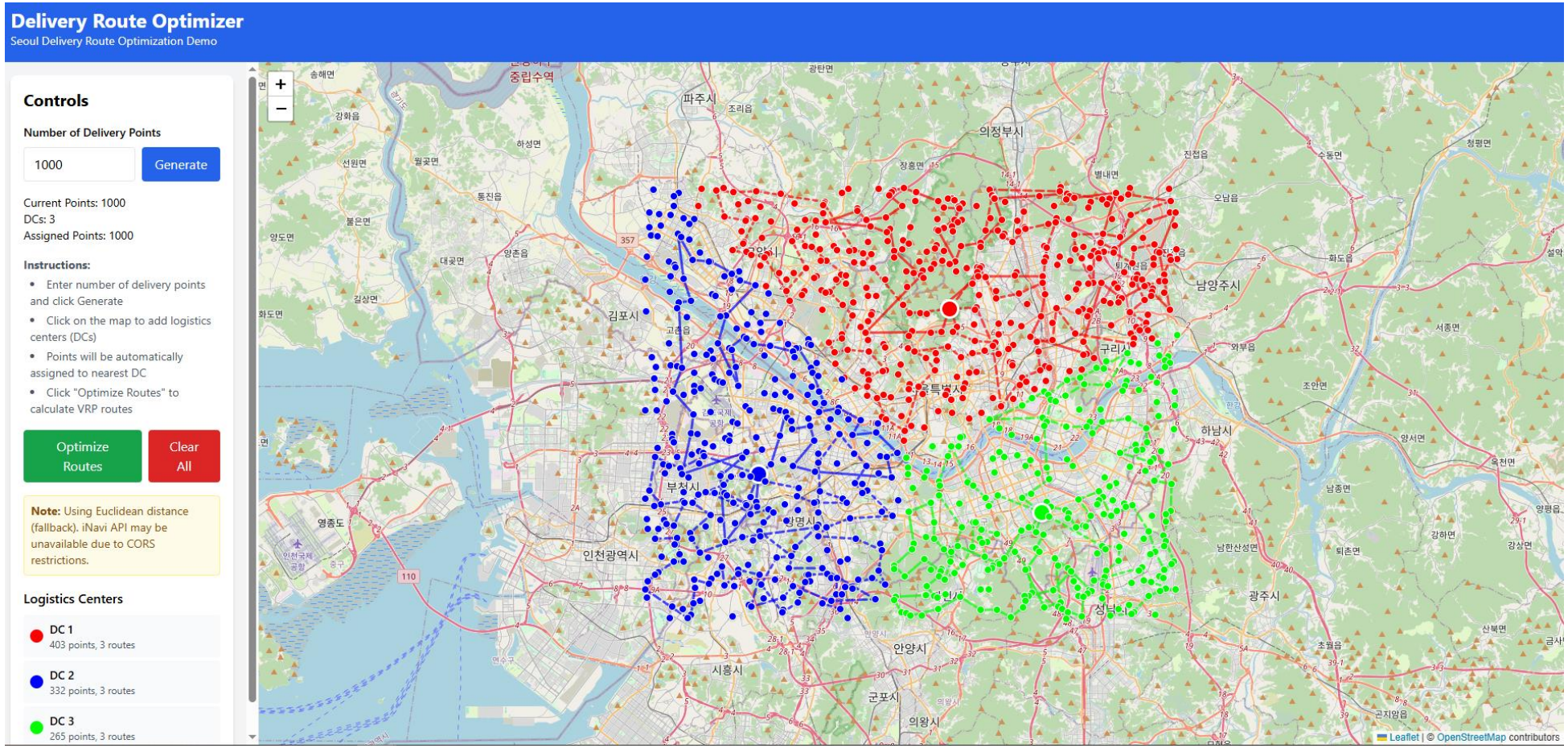
- Demo Browser



Demo

Example: Seoul Delivery VRP Demo)

- Demo Browser



Evaluation Criteria

Total: 40 points
Submission Files: Project Report (pdf) / Source Code (zip file)
Team: 1 to 4 (For more people, we anticipate more quality)

- **Problem Definition (10 pts)**
  - Clarity, realism, and justification of the decision-making problem
- **Optimization Modeling (10 pts)**
  - Quality of Routing API input design (vehicles, stops, constraints, parameters).
- **Demo App Quality (10 pts)**
  - Functionality, UI/UX, stability, and completeness of the application.
- **Report (10 pts)**
  - Organization, readability, and insights of the report.

Evaluation Criteria

You can also do final project as an algorithmic development

Total: 40 points
Submission Files: Project Report (pdf) / Source Code (zip file)
Team: 1 to 4 (For more people, we anticipate more quality)

- **Problem Definition (10 pts)**
  - Clarity, realism, and justification of the decision-making problem
- **Optimization Modeling (10 pts)**
  - Correctness and completeness of the proposed method
- **Effectiveness of the Method on the problem (10 pts)**
  - Empirical evaluation or theoretical analysis on the proposed method
- **Report (10 pts)**
  - Organization, readability, and insights of the report.