MCI project First Milestone Report

Team number: 03
Project Title: A Map Management Platform for Self-Driving Cars

Milestone 1	Activities	Planned Outputs	Achieved Outputs
Restate the milestone from your Draft plan.	Restate the key activities from your draft plan.	Restate the planned outputs from your draft work plan.	Outline the actual outputs compared to what was projected (or type "same as planned")
The focus of our first milestone is backend. By the first milestone, the project team has been able to build a web-based application of map management for self-driving cars with basic functionalities, including: • fetching all the images and data from Mapillary using an API, • storing the data in an SQL database, • running algorithms to provide key information to the users i.e. nearby images for particular coordinates, • presenting the information through a website, • rendering the layers in the map	Setting up frontend using React	The frontend application is able to present the data on a website.	Same as planned
	Setting up Mapillary API integration with frontend	The frontend can fetch all the images and data from Mapillary using an API.	Same as planned
	Setting up backend environment with Django	The backend environment is set up with the administration panel.	Same as planned
	Visualising data from Mapillary API using Deck.gl	Data fetched from Mapillary API can be visualised with different layers.	Same as planned
	Designing database structure	Database design is visualised using a UML software for development and maintenance purposes.	Same as planned
	Fetching data from Mapillary to Django to be stored in an SQL database	Data from Mapillary including users, sequences, geographic data and images can be retrieved from Mapillary and saved to the backend database.	Same as planned
	Developing diagrammatic architecture explaining flow and hierarchical structure	A basic diagram is created to show an overview of the web-based application architecture.	Same as planned
	Developing algorithms to find images close to particular coordinates and having the same direction	The algorithms are able to retrieve nearby images for particular coordinates and images of cars driving in the same direction.	Same as planned
	NEW: Analysing different nearest neighbour algorithms and comparing with the currently implemented algorithm (KD Tree)	A better solution in terms of accuracy and performance for the KD Tree algorithm is found and implemented (if have).	Different nearest neighbour algorithms such as Ball tree and Quad tree have been analysed and benchmarked. Quad tree is proved to be a more efficient algorithm than KD Tree. Implementation for Quad tree is in progress under "Backend-Direction" branch.

Team reflection on progress	Provide some comments below regarding the completion of this milestone specifically around: 1. How is the project progressing? 2. Are there any differences between projected and actual outputs/outcomes?

1. Project progress:

- Overall, the team has managed to deliver most of the milestone 1 activities.
- We spent the first two weeks laying the foundation for the project, including:

- Setting up weekly meetings with the client and trying to understand the project requirements.
- Agreeing on a fixed schedule for the team's internal meetings and the communication channels i.e. Discord and Zoom.
- Researching to decide the frameworks and the type of database to use for the project.
- Getting familiarised with the chosen frameworks:
    - Backend: Django
    - Frontend: ReactJS
    - Database: SQLite
    - Data source: Mapillary
    - Map rendering: Mapbox
    - Layer rendering: Deck.gl
    - Version control tool: GitHub
- Setting up boilerplate code for backend and frontend.
- Preparing for administrative tasks i.e. using Trello for task management, finalising the templates for meeting minutes and agenda, using a rotation log to keep track of task allocation related to meeting minutes taking and agenda preparation.

- Since the third week, we have focused on developing functionalities for the platform as well as delivering the tasks required as per the course assessment.

Our achievements so far can be listed in the following categories:

- Frontend: setting up frontend using React, visualising data downloaded from Mapillary API using Deck.gl.
- Backend: setting up backend using Django, setting up Mapillary API integration with backend, fetching data and downloading images to be stored locally in the SQLite database, implementing the nearest neighbour algorithm KD Tree to find nearest neighbours in 100m radius of the queried coordinates and having the same direction.
- Project documentation: database relationship diagram and solution architecture.
- Course assessment: pitch presentation, business case and first milestone plan.

2. Differences between projected and actual outputs/outcomes:

- We planned to set up Mapillary API integration with our frontend to fetch data directly from the beginning of the project. However, after our research and tests, we realised that we had to write a function in backend to retrieve the data instead. The reason was that Mapillary API used the pagination method to divide their resources. When requesting a collection of resources from Mapillary API, we could receive only a limited number (a page) of resources per request. To navigate a collection, we had to send multiple requests and our requests must follow the links provided in the Link header in the previous responses.
- There was an additional activity suggested by our client during the first week of mid-term break (after we had already submitted the 1st milestone plan): to analyse different nearest neighbour algorithms and benchmark as well as implement them to compare with the currently implemented algorithm (KD Tree). This was a result of the problem we encountered with the KD Tree algorithm in terms of performance. Although the algorithm helped us to retrieve nearby images to proceed with developing other functionalities for the platform, we were concerned about the long time it took to query the tree from the large datasets with more than 3 million entries and then calculate K-Neighbours for all the data points (Geo-Coordinates). The client then advised us to research other possible solutions. We followed the instruction and found out that Quad tree was a more efficient algorithm than KD Tree. To prove that the new algorithm is a good fit for the project, we have to replace KD Tree with the new algorithm. However, due to some technical difficulties in integrating the new algorithm into the project code, we haven't been able to deliver this task in time for the milestone 1 deadline. We will continue working on this activity during our 2nd milestone.

Team reflection on managing problems	Have you encountered any problems to date? If so, how have you managed them?

1. The first problem we encountered was that all the team members were inexperienced in the modern web development frameworks used in the project i.e. Django and React. Python, Javascript, and HTML are the languages used in these frameworks and all of us had no to very little experience in writing codes in these languages. Therefore, it took us a lot of time to develop a new function or debug an error while having to deliver new tasks assigned by the client every week.
→ To solve this problem, we encouraged everyone to self-study by taking online courses, doing research and practising coding regularly. From the beginning, we also divided the team into two groups: frontend and backend to distribute the tasks evenly. Then, we gradually merged the groups to allow for knowledge sharing among team members and ensure that everyone was on the same page with the project implementation progress and also could back up for each other.

2. The second problem we encountered was that when we downloaded images from Mapillary using the provided URLs and saved them to our database, it took very long to download all the images. The estimation for downloading 3 million images was around 30 days with the initial implementation.
→ To overcome this issue, we researched several solutions and implemented the solution provided by an open-source project called Excel Parser Processor which allowed us to reduce the downloading time to 50 hours.

3. The third problem we encountered was the lack of communication from the team member Aryaman Dhawan. Aryaman used to miss our internal team meetings quite often without reasonable notice and also fail to deliver some of the tasks we assigned to him on time without a valid reason.
→ We discussed this issue openly during the team's internal meetings and gave Aryaman some warnings. After that, we noticed some improvements in his performance so we decided to keep monitoring the situation. If the situation got worse and we were unable to solve this conflict internally, we would need to escalate the issue to the course coordinator as per the standard conflict resolution mechanism.

Supervisor assessment	Please, rate your team (1) effort, (2) project progress and (3) their self-reflection for milestone 1 Rating scale 1-10 as per standard marking scheme, ie 5 is a Pass and 7 is a credit. Add some comments to explain your rating
Effort: Progress: Reflection:	