MCI project Second Milestone Report

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

Milestone 2	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")
By the second milestone, the project team has been able to build a web-based application of map management for self-driving cars with most of the required functionalities, including: • Quadtree based algorithm implementation, • backend-frontend integration, • job scheduler • frontend functionalities i.e. choose a region and a layer, retrieve nearest neighbour images, filter data by date range, and change the colour of the clicked object and its sequence, • user-friendly interface for frontend. The project team has also completed other activities i.e. • testing, • bug fixing, • documentation.	Implementing Quadtree based algorithm for backend	Quadtree based algorithm is successfully integrated into the project code and completely replaces KD Tree algorithm.	Same as planned
	Integrating backend with frontend, including allowing frontend to query data from the database set up in the backend	Backend and frontend are successfully integrated through the API built via Django REST framework, and data in the backend can be queried from the frontend.	Same as planned
	Populating the SQLite database with Mapillary data automatically every two weeks	A batch job is set up successfully and scheduled biweekly to automatically save new data from Mapillary to the database.	Same as planned
	Developing functionalities for frontend including: - positioning the map based on the chosen region - choosing different layer types - retrieving neighbour images having the same direction when an image point is selected - filtering sequences by user-defined time intervals changing the sequence's colour of the selected point	The frontend has a user-friendly interface with all the required functionalities.	Same as planned
	Performing unit testing (UT) & bug fixing (if have)	UT passes with the bugs found by the team fixed.	Same as planned
	Preparing project documentation including technical specifications and user guide	The required documents are well presented to help the client easily use, maintain and enhance the platform.	Same as planned

	Supporting user acceptance testing (UAT) & bug fixing (if have)	UAT passes and other functionalities suggested by the client to improve the application are implemented.	Same as planned
	Publishing the website	The website is successfully published.	The team have managed to publish a test version of the website via Heroku and it worked smoothly with the free Heroku account, including two functions: adding a new region via Django admin site and navigating to a new region on the map. There are several items to pay for when deploying the website on Heroku e.g. Heroku dynos, Heroku add-ons (Heroku Postgres for database and HDrive for images) and Amazon S3 bucket so the team have to wait for the client's decision before proceeding.
	Supporting post-go-live activities	All the issues highlighted by the client (if have) are fixed and source code is shared with the client on GitHub.	Source code has been shared with the client on GitHub. Since we haven't published the website, no post-go-live activities have been carried out.
	NEW: Implementing parallel download mechanism for downloading images	Images are downloaded using multiple processing and the image downloading function is integrated into the job scheduler.	Same as planned
	NEW: Developing additional functionalities for frontend: zoom and loading.	The new frontend functionalities work as expected.	Same as planned