Development Plan
Park'd

Team #29, caPstOneGroup
Albert Zhou
Almen Ng
David Yao
Gary Gong
Jonathan Yapeter
Kabishan Suvendran

April 5, 2023

Table 1: Revision History

Date	**Developer(s)**	**Change**
Sep 22, 2022	Albert, Almen, David, Gary, Jonathan, Kabishan	Revision 0
Nov 2, 2022	Almen	Adding to Technologies

Contents

List of Tables

This document is the Development Plan for the project, Park'd, an application that aims to help the average driver quickly find an open parking space, especially on busy days with few spots available.

1 Team Meeting Plan

The following is the plan for when, the location, frequency, and duration of the team meetings:

Where	When	Duration
H.G. Thode Library	Every Wednesday at 10:30AM	1 Hour
Discord	Every Monday at 9:00PM	2 Hour

Table 2: Team Meeting Plan

1.1 Meeting Roles

The table below, Meeting Roles, identifies the assignment of roles during the meeting.

Meeting Role	Name
Participant	Everyone
Chair	Jonathan Yapeter
Scribe	Albert Zhou
Timekeeper	Albert Zhou

Table 3: Meeting Roles

The responsibilities of the meeting roles identified in Meeting Roles are as follows.

1.1.1 Participant

1. Understand the agenda and the purpose of the meeting
2. Propose topics prior to the meeting that needs to be discussed
3. Contribute to the discussions of all agenda items

1.1.2 Chair

1. Guides the group through the predetermined agenda
2. Facilitates respectful discussion among all participants
3. Establish equal opportunities to speak
4. Clarify conclusions and next steps at the end of each meeting

1.1.3 Scribe

1. Create the meeting minutes prior to the meeting
2. Notify all participants of the agenda before the meeting
3. Note all the decisions, conclusions, and action items presented during the meeting
4. Compile notes in a standard template with consistent formatting

1.1.4 Timekeeper

1. Monitors, records, and manages the time it takes to accomplish an item in the agenda

1.2 Rules of Agenda

The following are the rules of agenda for all in-person and online team meetings:

1. An agenda must be created prior to every meeting. The following guidelines should be followed when creating the agenda:
    - First topic should be to review the agenda
    - An estimated allotted time for each topic
    - Topics must be formulated as questions
    - Identify the people responsible for each topic
    - Seek input from all team members
2. All members must be present during the meeting and attendance will be recorded
    - Scheduled absences must be communicated to the rest of the team at least one day in advance of the meeting
3. Meeting minutes will be taken during every meeting by Albert, our Scribe.
4. Prior to current agenda, any conflicts and/or discussions from the previous action items must be addressed
5. All members must work and communicate respectfully to each other during the meeting

2 Team Communication Plan

The meetings mentioned in Team Meeting Plan will be conducted in-person at H.G. Thode Library on campus or on Discord. Members are to be seated together in the first floor of Thode or in the appropriate voice channel by the start of the meeting.

To facilitate communication outside of designated meeting times, a Messenger group chat with all members of caPstOneGroup has been created. The expectation for all team members is to check the Messenger group chat at a minimum of twice per day and will respond to

mentions at their earliest convenience. Any urgent issues that arise regarding milestones shall be brought up immediately in the Messenger group chat.

Git issues will be created with the appropriate Priority/Severity label and assigned to the team members involved if there are any tasks needed to be done or bugs needed to be fixed. Discussions regarding the project code will be predominantly done through these Git issues, however, can be reinforced in the Messenger group chat.

Name	**Discord**	**macid**	**GitHub**
Albert Zhou	Ferman	zhouj103	albertzevanescent
Almen Ng	Mumbojumbo	nga18	almen-ng
David Yao	d i n g u s	yaod9	davidwyao
Gary Gong	Shawnty	gongc12	GaryGong27
Jonathan Yapeter	yapeter	yapetej	JAYapeter
Kabishan Suvendran	midnightmarauder	suvendrk	Kabishan

Table 4: Team Meeting Plan

3 Team Member Roles

The table below, Member Roles, identifies the assignment of main roles of the team. These roles are not fixed and can change when the need arises.

Name	**Role(s)**
Everyone	Developer QA Tester DevOps
Albert Zhou	Scribe
Almen Ng	Project Lead Team Liaison
David Yao	Documentation/LaTeX Expert
Gary Gong	Machine Learning Expert
Jonathan Yapeter	Git Expert & Maintainer
Kabishan Suvendran	User Interface Expert

Table 5: Member Roles

3.1 Team Member Role Descriptions

This subsection will provide descriptions on all the team member roles identified and assigned in Member Roles.

3.1.1 Developer

- Design and builds application features
- Produce clean and efficient code based on specifications
- Troubleshoot and resolve any bugs that may arise
- Propose and execute improvements to the code base
- Provide technical documentation to all code for reference

3.1.2 QA Tester

- Test new and existing features
- Automate unit and integration tests
- Report any bugs or failures and suggest fixes

3.1.3 DevOps

- Automate repetitive tasks through GitHub actions/pipelines to decrease the workload of team members
- Facilitate performance testing and benchmarking to evaluate how reliable the system runs
- Introduce processes, tools and methodologies to support the software development lifecycle

3.1.4 Scribe

- Refer to Scribe in Meeting Roles for the responsibilities of a Scribe

3.1.5 Project Lead

- Ensure the team is on track with completing all milestones outlined in the capstone course outline by the deadline
- Devise plans that support project goals
- Address any bottlenecks or conflicts that may arise

3.1.6 Team Liaison

- Disseminate information given by the instructors to the rest of the team

3.1.7 Documentation/LaTeX Expert

- Formats LaTeX documents for milestones with written reports
- Fix any compilation errors regarding LaTeX documents that may arise
- Create new LaTeX commands that may be needed
- Create and enforce a standard for documenting code and features

3.1.8 Machine Learning Expert

- Answer any questions related to Machine Learning, like Computer Vision
- Lead the building, training and deploying of models necessary for the application

3.1.9 Git Expert & Maintainer

- Answer any questions related to Git, like PRs and Issues
- Track all ongoing Git issues/PRs/tasks and make sure that they are reviewed and merged in a timely manner
- Ensure that all changes made in the Git repository are following correct naming conventions, coding guidelines, and follow the proper workflow as mentioned in Workflow Plan.

3.1.10 User Interface Expert

- Lead the design of the application to support user interface and user experience principles

4 Workflow Plan

We will use GitHub to maintain the code, the documentation for said code and the milestone deliverables, including this report. All of the code or documentation that will be viewed and marked by the teaching assistant will be merged into the main branch before each milestone's due date. Changes will not be committed directly onto the main branch, since the main branch will be protected from erroneous pushes. Instead, each developer will create their own branch, create a PR and assign another team member to review and approve their changes before they are merged into the main branch. For changes related to the code, in addition to team approvals, the code must also pass all of the unit tests that were set up as a part of the Git Workflow, and Pylint will be used for source code analysis to detect any errors and enforcing coding standards. For changes related to LaTeX files, the files must compile and produce a PDF document before the changes are merged.

To manage issues, including template issues, issue classifications, and general inquiries about certain aspects of the code, our team will employ a project management tool, such as

Zenhub, on which, we will use a project board to create and manage issues. On this board, we will sort each and every issue into several categories:

- Untriaged (not assigned a priority or time frame for completion)
- To Do (work that is yet to be worked on)
- In Progress (being actively developed)
- Code Review (code completed and PR opened)
- Done (code approved and merged into main)

We will also use labels and tags, such as front-end, back-end, documentation and machine learning, to indicate which issues correspond to which component of the project. Using GitHub Issue tracker, we will reference specific issues, tag developers to notify them of the issue, set up meetings to discuss the issue, and finally link the PR that resolves the issue.

As a part of our Workflow Plan, we will maintain a Contributor's Guide, in which, we will mention the specifics of pushing quality code and documentation. For example, each function, method and class should be documented based on their parameters, purpose, and return value. The exact formatting of the documentation will be based on the style guide for the chosen programming language. Furthermore, the documentation must be succinct, coherent, and must not contain spelling or grammar mistakes. This is also required of LaTeX documents, as well as project deliverables. Furthermore, another aspect of the Contributor's Guide is how pull requests are opened. Each pull request must be opened with a description that mentions what the code changes will do and it should be assigned to at least one other team member for approval.

5 Proof of Concept Demonstration Plan

The Proof of Concept (POC) demonstration plan highlights the risks we have discovered in the creation of our project, as well as how the demonstration should show that those risks can be overcome.

5.1 Risks

- **Finding a suitable parking area to test**: A parking lot administrator will need to agree to either make their video equipment available for use in the program, or allow such equipment to be installed. A video feed is integral to the proposed functioning of our program, because of the need for machine vision in analyzing parking activity.
- **Computation time**: Machine learning is computationally expensive, so a poorly optimized program may not respond with the promptness required for a real-time driving aid platform.

5.2 POC Demo

Our proof of concept will demonstrate the program's capability to detect the state of a parking space. It will be a recorded run with a single parking spot. The program will first poll the state of the spot while empty, and then a vehicle will pull into the spot once the correct status has been determined. The demonstration's relation to the given risks is as follows:

- **Finding a suitable parking area to test**: Our need for a suitable parking spot will be fulfilled by this time if the POC demonstration is to go ahead as planned. If no such place is found, some group members will volunteer their driveways as simulated parking spots. In addition, given our program's usage of video feeds, we are also prepared to proceed using publicly available online feeds of parking lots.
- **Computation time**: Only the baseline functionality for our program, the detection of parking spot occupation, is being tested as part of the POC demonstration. By showing a baseline for the computation time that we should expect from this task, we will then decide what additional features we can afford to add from our stretch goals. If even the baseline task is too intensive, we will explore either moving the back-end to a more powerful machine, or finding a non-machine learning algorithm to accomplish the task.

6 Technology

The following are the technologies we will be using for our project.

- Programming languages
  - We will be using Python for training the machine learning model as well as making a back-end service
  - JavaScript and HTML CSS will be used for developing the front-end web page of our project
- IDE
  - Front-end: Visual studio code
  - Back-end/ML: PyCharm
- Linter and Code coverage measure tools
  - We will be using Pylint and flake 8 for different purposes including check for errors, tries to enforce a coding standard, looks for code smells
  - Coverage.py would be used to measure the code coverage for Python programs
  - Istanbul will be used to measure the code coverage for UI test cases with Cypress.
- Testing

  - Pytest will be used for making unit test cases for Python programs
  - Cypress will be used for making UI test automation.
  - Postman API Platform will be used for making API test automation.

- Libraries
  - OpenCV,TensorFlow Numpy will be used for training neural networks and configuring training data sets.
  - Flask will be used for creating back-end services of our project

- Performance measuring tools
  - Not applicable

- DevOps
  - Basic DevOps plans including static code analysis, linting, running unit test cases using Pylint and Pytest will be conducted when pull requests are created.

- Documentation
  - Doxygen will be used for generating documentations for the back-end services and APIs
  - Development document will be edited through overleaf LaTex editor in a collaborative fashion

- Hardware
  - Cameras from the parking lot

7 Coding Standard

The back-end part of the project and the algorithm part for training the machine learning model will follow the PEP8 python coding standard. The linter in the workflow plan will ensure the standards are followed for the code written.

8 Project Scheduling

Project scheduling is based on the course deadlines and will be tracked using Zenhub, a project management tool that is directly integrated into GitHub through an extension. Zenhub's roadmap tool displays a timeline and blocks during which given tasks should be completed. Roadmaps will also integrate data from our issues. Zenhub also has a counter feature that allows you to approximate an estimated amount of time it takes to complete an issue which would help us get a better idea of what our timeline would look like.

To determine when a major milestones are due, we will take the course deadlines, discuss approximately how much time it will take to complete each task, and define a definitive date

to complete the major milestones by while allocating some time prior to the actual course deadlines for any spillover.

For decomposing of larger tasks into smaller ones, we will discuss what is needed for those tasks, write out issues for each individual task, then create a Zenhub epic, which is a way to group all related sub tasks represented by issues into one overarching issue.

In terms of assigning tasks, we will have a discussion on who is best suited to accomplish the task while also providing opportunities for other members to contribute based on their interest. Another factor that need to be considered is the amount of time each person has per week to accomplish a task. Using the Zenhub counter feature, we will be able to divide the tasks up so each member will spend around 9-12 hours a week for the course. For example, if a task involves computer vision and it would take 9 hours to complete, the Machine Learning Expert will only be assigned this task and potentially one smaller task that takes less than 3 hours.