|  |  |  |
| --- | --- | --- |
| **Name:** Erik Meurrens, Benjamin Simonson, Ryan Jalloul, Evan Tobon, Samer Khatib  **Date:** 10/29/2024  **Team:** Parking Availability System (PAS) - ParKings | | **Upcoming Milestone:** Design Prototype  **Stakeholder:** Raven Kim Ramos |
| **Goals for upcoming milestone** | Produce a list of tangible goals for the upcoming sprint. Include a title and brief description of the goal, an estimated allotment of time, and a description of how it relates to the upcoming milestone specification. | |

* View video stream on Raspberry Pi through hardwired connection
  + Story points: 5
  + While we can capture a single image at a time, we need to verify that we are able to get a live image stream from the camera module, as well as have a way to manipulate the image capture rate or frame rate.
* Provisional object detection model on RPi
  + Story points: 8
  + It is time to start working on the RPi’s object detection algorithm. We need to be able to detect cars passing into and out of the parking facility visually in order to transmit an image frame to the ALPR model on the cloud.
* Provisional ALPR model on AWS
  + Story points: 8
  + To identify cars and their characteristics, a more complex model needs to be run that might be too much for the RPi to handle with its limited hardware (1.4 GHz processor, 1 GB RAM). The ALPR model will run on AWS, and the RPi will capture an image frame of a detected car to send to the AWS model to extract the license plate number.
* Design and Print Models for Hardware Housing
  + Story points: 8
  + We need a housing to compartmentalize all the components of our hardware system: the Raspberry Pi, Camera Module, Power bank, sensors, and potentially a motor.
* Configure EZ Park App to access SQL database
  + Story points: 5
  + Now that we’ve gotten access to the EZ Park app codebase from the developer to access UF parking facility information, we can use the hard-coded data in the app to populate the LOT database table. This information is necessary so we can be able to track the capacity of each LOT entry.

|  |  |
| --- | --- |
| **Achievements since former milestone** | Report on the status of your progress since the last worksheet was filled out. Include a description and explanation for why each target was or was not accomplished. Estimate the time spent on each. |

* Completed Pre-Alpha Build
  + Completed provisionary API to access database
    - Accomplished (15 story points)
    - Multiple tasks from the Kanban board were combined into this item
      * “API Get Car endpoint”
      * “API Save Car Endpoint”
      * “API Delete Car Endpoint”
      * “API Update Car Endpoint”
      * “API Get all Cars Endpoint”
    - Necessary to streamline database access from client devices (mobile app, RPi)
  + Connect camera module to RPi
    - Accomplished (1 story point, ~ 1 hour)
    - Simply attaching the connector from the camera module to the board
  + Get image capture frame from camera
    - Accomplished (3 story points, ~ 3 hours)
    - Ensures that the camera module is properly connected, and we can begin tasks that require image data as input
  + Establish connection from App to PostgreSQL database
    - Accomplished (3 story points, ~ 4 hours)
    - The app needs to be able to access the real-time parking availability data from the PostgreSQL database in order to display that data to users. Given the code hasn’t been touched in a few years, and a relatively obscure framework, Flutter, is being used to program the app, this ended up taking a while to complete. However, in the end, a connection was able to be established between the app and the database.
  + Establish connection from RPi to PostgreSQL database
    - Accomplished (5 story points, ~ 5 hours)
    - The RPi needs to be able to access the database in order to identify cars passing through its field of view to count car throughput and increment the real time parking availability.
  + Prepare app repository
    - Accomplished (2 story points, ~ 2-3 hours)
    - We need to put the codebase for the app in the repository for us to access and program. This required finding the old app files, verifying it still functions as expected, removing code with API keys and other secret credentials, and adding the app files into a folder within the repository.
  + Install VPN client on RPi
    - Accomplished (Listed in contribution log, ~ 2 hours)
    - One issue with accessing the board remotely is that it always needs to be connected to the UF network; on the weekend, this becomes a problem because the power bank powering the board can deplete, and we won’t be able to access the lab after hours to replace it. Therefore, it became important to make sure the RPi can connect to the UF WiFi network with a VPN, so that we can still remotely access it.