CS 3354 Software Engineering

Final Project Deliverable 1

Smarter Parking for a University Campus

Patrick O'Boyle

David Hiser

Gaurang Goel

Saumya Karia

1. Final Project Draft Description

Good choice for a topic! Watching movies is such a popular activity, particulary during these times. Your design is promising to be a very useful one for determining a good one to see, as there are so many of them.

Your team should consider a more detailed break down of the tasks as there are more of them (please see the project specs). There is no need to resubmit this proposal, though. Just make sure that you don't miss any task. Also include the "had to be added late" new member to your team.

In the final report, please make sure to include comparison with similar applications -if any-, make sure that you differentiate your design from those, and explicitly specify how.

Fair delegation of tasks.

Please share this feedback with your group members.
You are good to go. Have fun with the project and hope everyone enjoys the collaboration.

Complying with Changes:

- More detailed tasks: We provided a detailed breakdown for tasks in part 3 and will continue to update it for future deliverables
- Late member: They are from a different section, so that news didn’t apply.
- Comparison with similar applications + differentiation: We’ll make sure to cover this in our final report.

2. Github:

Link: https://github.com/saumyaK512/3354-Smart_Parking

Split tasks:

- 1.3: Completed by Saumya
- 1.4: First was accidentally completed by Saumya. Refer to the new commit by Gaurang.
- 1.5: Completed by Patrick

Note on the 1.4 mistake: Retroactively messing with the Git history is a bad idea. It's better to layer new commits over old ones as we did.

3. Tasks (Deliverable 1)

Patrick O'Boyle:

- Setup document (done)
- Part 1 (done)
- Part 2 (1.5) (done)
- Part 9 (done)

David Hiser:

- Part 6 (done)
- Part 7 (done)

Gaurang Goel:

- Part 2 (1.4) (done)
- Part 8 (done)

Saumya Karia:

- Part 2 (1.2,1.3) (done)
- Part 4 (done)
- Part 5 (done)

4. Software Process

We have chosen to employ the Spiral model as our software process model for this project. The project's overall size and complexity are the major factors. It makes intuitive sense to adopt the spiral approach since we are developing an entire IoT system that will communicate with various camera hardware and sensors throughout multiple parking structures.

Although this project is currently only focusing exclusively on the University of Texas at Dallas campus, if it needs to be expanded to multiple campuses across the country, the spiral model is the most practical option because it combines the controlled and systematic aspects of the waterfall model with the iterative nature of prototyping. Since no two campuses are the same, as the project grows, different stakeholders will have different requirements. For this reason, being able to prototype iterations quickly is very beneficial because it would only make sense to implement the project during the summer semesters when universities typically have fewer students and traffic on campus.

Additionally, this aids in developing a rapid prototype of the system that stakeholders can use to offer input, but which won't impact the whole campus if it breaks or needs significant adjustments.

5. Software Requirements

Functional Requirements:

1. Create a log file at the end of the day which includes the summary for each parking structure
2. Allow user to manually set occupancy for each parking structure
3. Check if cars license plate has been registered with a parking pass to allow entry into parking structure
4. Accurately deduce the number of parking spots occupied in a parking lot based on the parking category
5. Parking occupancy of each parking structure should get updated accurately based of a car occupying a parking spot or leaving the parking spot.
6. Check if a car is parked in the correct parking spot based on the type of parking pass linked to its license plate

Non-functional Requirements:

1. Product Requirements:
    a. Usability Requirements: The user should be easily be able to enter the garage without actually physically having use the system since everything is done by the sensors and cameras. Additionally the app should show the show the users the capacity for all the parking lots and a combined capacity of the campus
    b. Efficiency Requirements:
        i. Performance Requirements: The software needs to run 24/7 since the parking garages are available all the time. For peak hours the system should be able to respond within 5 seconds of an availability of the parking lot or if a space is occupied, this time would be our 99th percentile. For the rest of the time it would be acceptable if the system takes upto a minute to respond. For the phone software, it should be able to run on android OS 7 and upwards and for IOS 10 and above.
        ii. Space Requirement: In terms of the space the software should not take up more than 100 mb on the users phone, for the server we would require a bare minimum of 250 gb of storage with 16 gbs of ram space and 8 CPU Cores, all of this would include the storage space for the database of each parking space, ubuntu and other dependencies such as dockers, kubernetes etc. Additionally since the software uses some level of machine learning its good to have a good processor in the server.
    c. Dependability Requirements: The Worldwide Electrotechnical Commission (IEC) sets and maintains international standards through its Technical Committee TC 56, which provides systematic methodologies and tools for assessing and managing the dependability of equipment, services, and systems throughout their life cycles. Therefore this system should be able to hold up to those standards.
    d. Security Requirements: This system should hold up tot the security standards created by the OWASP Application Security Verification Standards.
2. Organizational Requirements:
    a. Environmental Requirements: The servers should be housed in a dry, clean, well-ventilated, and temperature-controlled environment. The client should perform tests provided by the ASHRAE report, “Thermal Guidelines for Data Processing Environments” on a regular basis in order to maintain the servers and

get the most performance out of the servers. The cameras and sensors used in the parking structures should also be cleaned on regular intervals and even be replaced if needed on a regular basis.

b. Operational Requirements: The system should follow the standards and protocols set by the NASA Systems Engineering Handbook and the Department of Home Land Security: Developing Operational Requirements Standards and should check for the following sections in the operational requirements:

    i. Electrical—Power consumption, efficiency, signal integrity

    ii. Mechanical—Strength, motion required

    iii. Structural—Capability to withstand environments in mission profiles

    iv. Optical

c. Developement Requirements: The Software would be a culmination of multiple softwares such as Python for the machine learning aspect and communicating with the different hardware, the device softwares would use either Javascript, Kotlin and Swift and the server could be managed on javascript with node.js and mongodb as its dependencies. Additionally the software development would follow the agile framework.

3. External Requirements:

    a. Regulatory Requirements: The software needs to comply with the following regulations:

        i. EU GDPR (General Data Protection Regulation)

        ii. GLBA (Gramm-Leach-Bliley Act)

        iii. PIPEDA (Personal Information Protection and Electronic Documents Act)

        iv. CCPA (California Consumer Privacy Act)

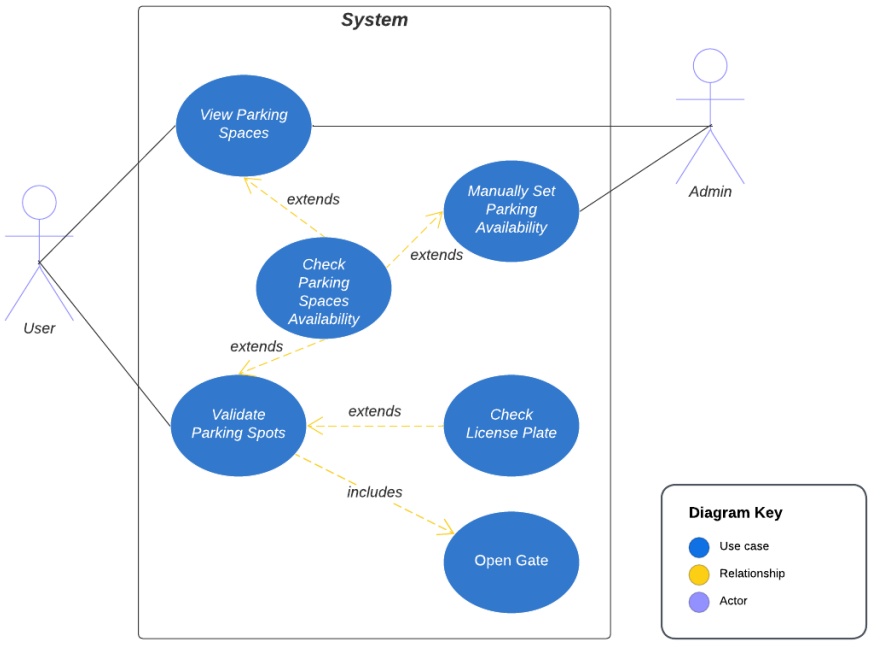
    b. Ethical Requirements: While developing the software and maintaining it, any one and everyone involved will follow the Code of Ethics set up by the IEEE-CS/ACM Joint Task Force on Software Engineering Ethics and Professional Practices.

c. Legislative Requirements:

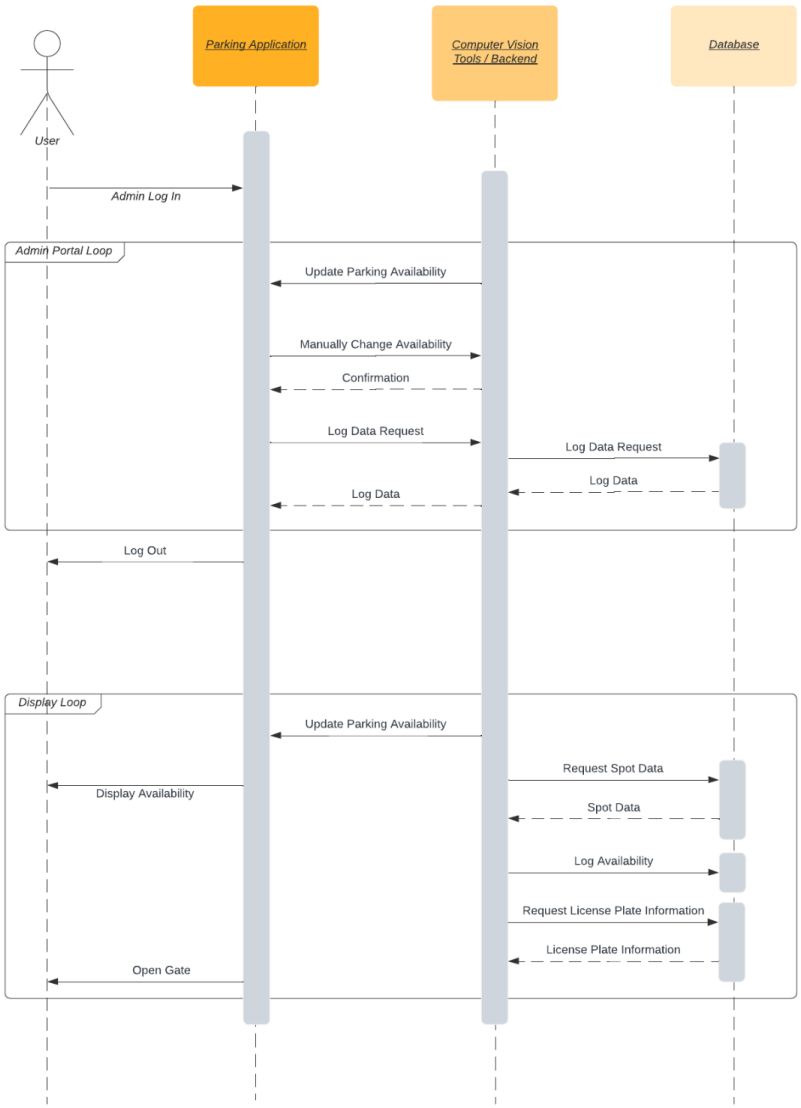
i. Accounting Requirements: The cost of the software development and anything else related to it should follow guidelines under US GAAP and GASB created by the US Government.

ii. Safety/Security Requirements: The development of the software should follow the standards for software safety created by IEEE SA - P1228

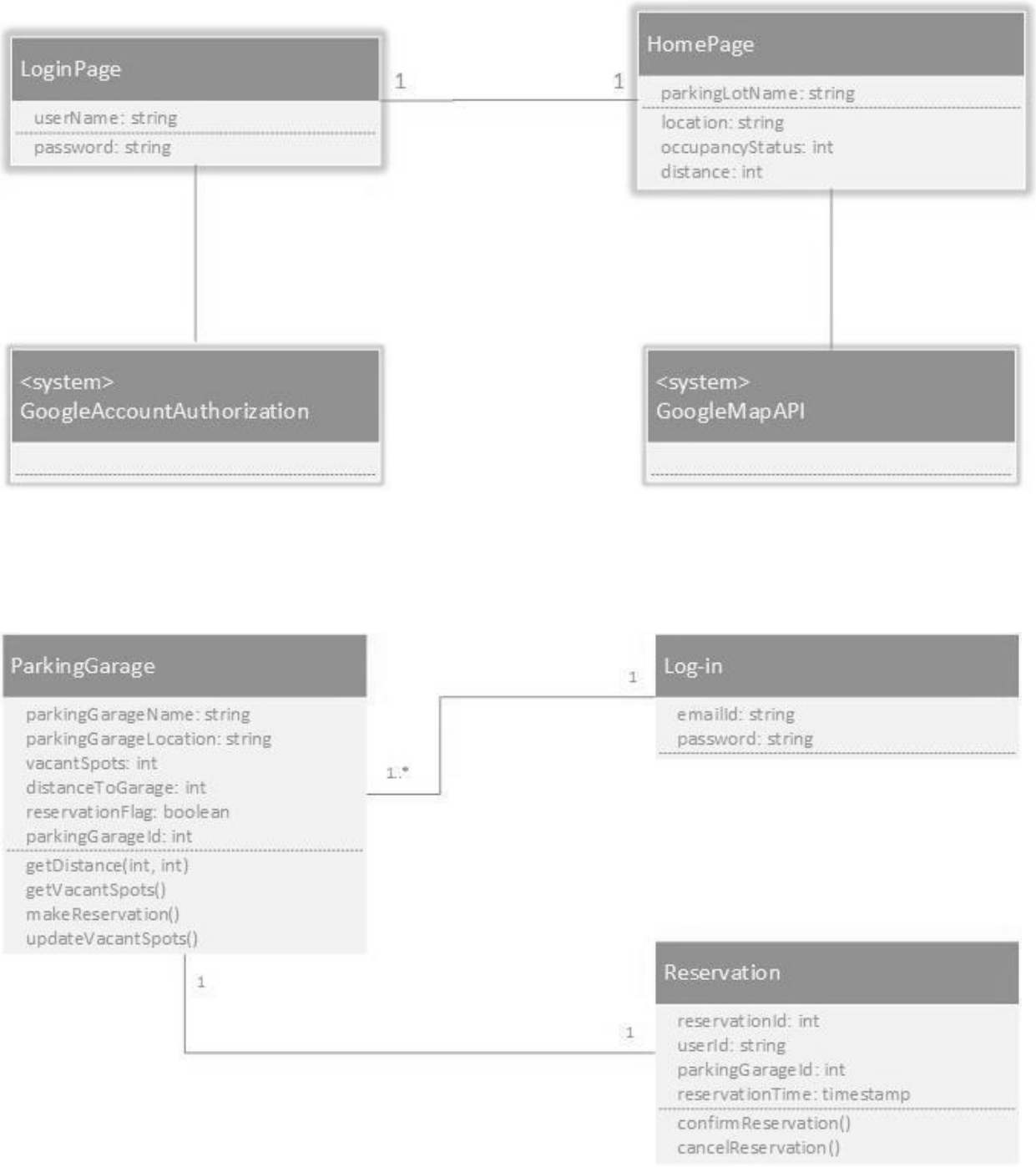
6. Use Case Diagram



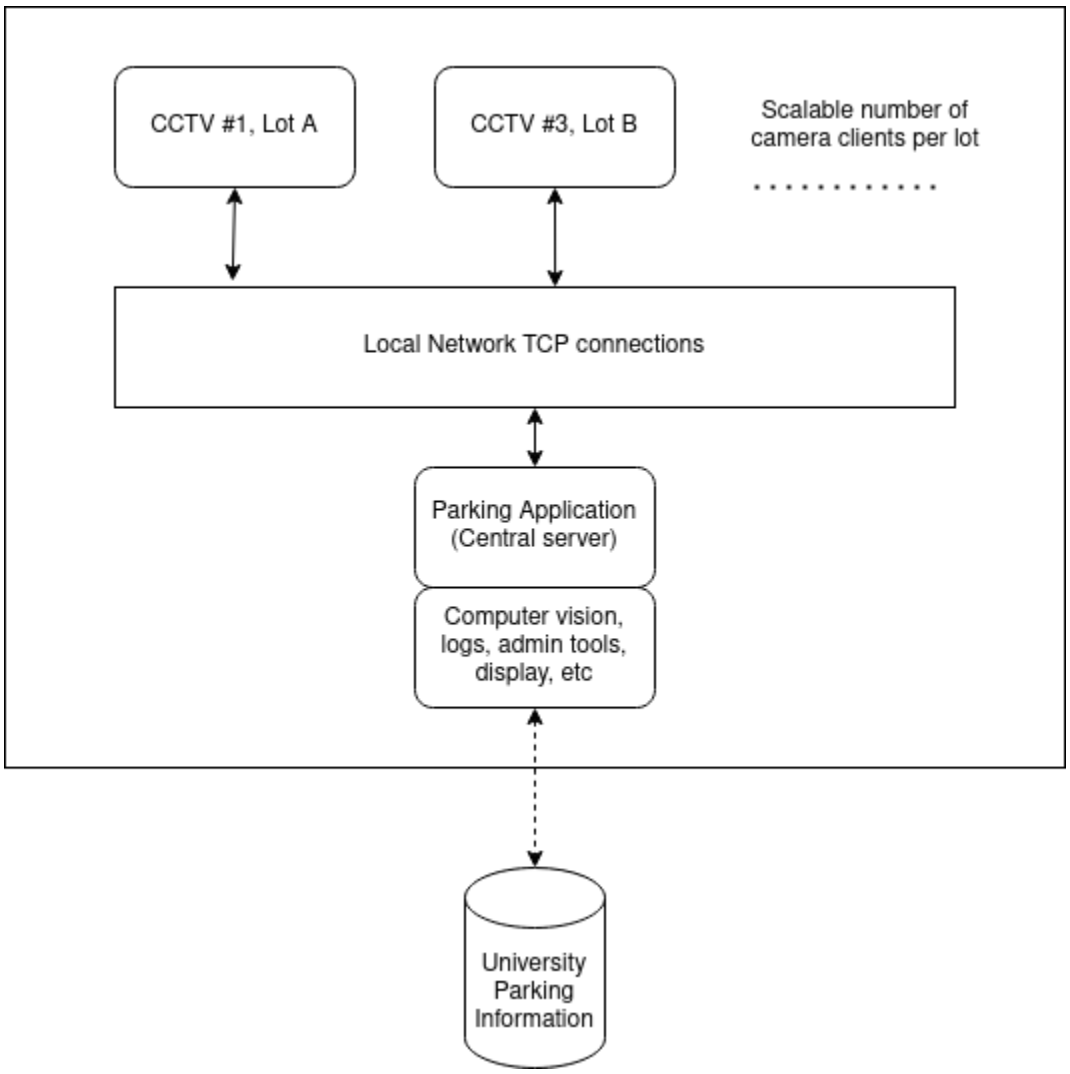
7. Sequence Diagram



8. Class Diagram



9. Architectural Design (Client-Server)



The above diagram is based on Figure 6.13 from the course textbook.

The bounding box defines the overall project. It is agnostic to the specific information provided by the given university, and can still operate without it.

10. Project Scope (also on Github)

Smart Parking Application

- Monitoring
    - Display occupancy information for each lot
        - tiled window displays for cameras at each lot
        - video feed, detected vehicles, car counts
- Administration
    - Ability to manually set/reset count at a lot
    - Access to log data and other relevant databases for parking information
- Logs
    - Collected at fixed intervals or whenever the count changes
    - Contains time stamps, occupancy count, license plates, parking pass ID's
    - Output an additional end-of-day summary log
- Parking Database
    - Need access to information to link license plates to parking pass types
    - Provided by university
- Object detection/tracking
    - Analyze video feed to detect and track cars and identify them within parking spaces
    - Cross-check license plates (or sticker passes) and parking spots
- Regulating Entry (if lot has a bar that opens/closes)
    - Only allow entry to lot/structure if the license plate is associated with the correct parking pass type for the lot/structure