DashCam Defender

Design Document

Team number 11
Professor Joseph Zambreno
Evan Timmons - Team Leader
Cobi Mom - Chief Mobile Engineer
Danny Yip - Test Engineer
Scott Vlasic - Head Hardware Concept Engineer
Durga Darba - Data Architect
Ismael Duran - Full Stack Engineer
sddec20-11@iastate.edu
https://sddec20-11.sd.ece.iastate.edu

Table of Contents

Executive Summary

Development Standards & Practices Used

Software practices

- Make code correct first and fast second
- Always test your code
- Agile Methodology
- Integration with hardware
- Simple design
- Pair programming
- Continuous integration

Hardware practices

- Integration with software
- Always test and make sure it is working
- Agile Methodology

Summary of Requirements

- Knowledge in mobile development
- Hardware development
- Database Design

Applicable Courses from Iowa State University Curriculum

- ComSci 309
- CPRE 288
- COMS 228
- SE/CPRE 185
- COMS 363
- CPRE 310

New Skills/Knowledge acquired that was not taught in courses

- Hardware Assembly
- Dart Programming Language
- Flutter Framework
- Hardware / Software integration
- API Calls

1. Introduction

1.1. Acknowledgement

We thank Dr. Joseph Zambreno for his consultation as our senior design advisor. We acknowledge the development done by our team members Evan Timmons, Cobi Mom, Danny Yip, Scott Vlasic, Durga Darba, Ismael Duran.

1.2. Problem and Project Statement

The roads as we know it are full of reckless drivers. What if there was a way to automatically detect when you are near a reckless driver? The solution that we are proposing is the ability for everyday consumers to come together and crowdsource information on reckless drivers. This is done by utilizing DashCam Defender, a product designed by us to automatically scan license plates and report reckless drivers when they are encountered instantly. Reporting is supported by a public facing website and app. This way, users can look up and report reckless drivers from multiple fronts.

1.3. Operational Environment

Since the product will be mounted on the windshield, it needs to be able to handle the fluctuation in temperatures that may occur in the car. Therefore, we will have to construct a container to protect the device from severe weather when the car is unattended as well as from consumer damages. This container must also be able to handle rough road conditions and keep the camera stable. Another condition we must consider is the camera's ability to capture data given poor weather. To ensure this, we will have to use a camera that returns images in 1080p or higher.

1.4. Requirements

Project requirements

- Dash camera that can record in 1080p
- A powerful enough computing board that can process machine learning application
- A computation board that can work with peripherals

Functional Requirements

- A license plate reader that can accurately read license plates from nearby cars
- A platform where the user can report and view information

UI requirements

- Have an aesthetically pleasing but simplistic application for the user
- Have a driver mode screen for the user to avoid accidents

1.5. Intended Users And Uses

We have three major intended users for our product:

- Police Departments
    - Police can lookup license plates and their last known locations to solve crimes faster
- Everyday Drivers

    - The everyday driver can report poor drivers and get alerts to notify them of when they are in the presence of a poor driver.
- Insurance Companies
    - Insurance companies can use driver ratings and dashcam footage to adjust their rates and claims

1.6. Assumption And Limitations

Assumption:
The end product will be able to work under a certain degree of poor weather. It will not be able to work if the weather is too bad, that the camera is not able to capture any clear picture. The end product will be used in the United States.

Limitations:
The end product will be a dashcam and a mini pc in the client's car, completing the Dashcam Defender apparatus. The cost to produce the end product shall not exceed one hundred and fifty dollars. The system will not require more than 12 Volts. (The most common voltage found in an Automobile Auxiliary Power Outlet)

1.7. Expected End Product And Deliverables

Hardware Product

- The expected physical product is that we have a dash camera and a computational board that can communicate with each other
- The computation board is integrate with ALPR (American License Plate Reader)
- The computational board can send data to a phone device

Mobile Application

- An aesthetic mobile application
- A driver mode screen
- The application should be able to send data to a server
- User should be able to view information like cars encountered

2. Specifications and Analysis

2.1. Proposed Approach

Our proposed approach is to find existing resources that will help us fill in for things that we need. A mobile application will be developed utilizing the Flutter Framework; We chose flutter because of its great layout design and versatility to create an android application and IOS application with one project. The camera and mini PC have to work together and the other half of the team will work on that with existing resources found. The mini PC we are going to use is the NVIDIA Jetson Nano, we chose the Jetson because a Raspberry Pi isn't powerful enough to do machine learning application. The camera we are going to use is a standard 1080p camera. Our solution to analyze license plates is to use an open-source American License Plate Reader; Developed by Rekor.

2.2. Design Analysis

Currently we have a mock database to test the connection between the hardware and the software. From here, we need to move into figuring out how to organize and send the information to fill the tables in the database. Our final version of the product will be doing this automatically, so we need to ensure the information can be processed efficiently. On the hardware side, our goal is to be able to read license plates at a rate of 5 per minute. On the software side, the program has to be able to multithread efficiently enough to process each license plate and store the information separately. The main strength of the project is in the database and the pushing and pulling of the data from it. The weakness, currently, is in the way we are pushing the data when the device doesn't have easy access to the internet.

2.3. Development Process

We are going to be following the Agile Methodology for our development processes. Specifically, we're going to split up into two separate teams of three; one team focused on software and the other focused on hardware. We think Scrum is the best choice for the development process because it will allow us to create small goals that are attainable as well as foster communication and collaboration within each group. We'll incorporate 2 week sprints because it aligns with the submissions of the bi-weekly reports. We will keep track of the work to be done in sprints in Trello.

2.4. Conceptual Sketch

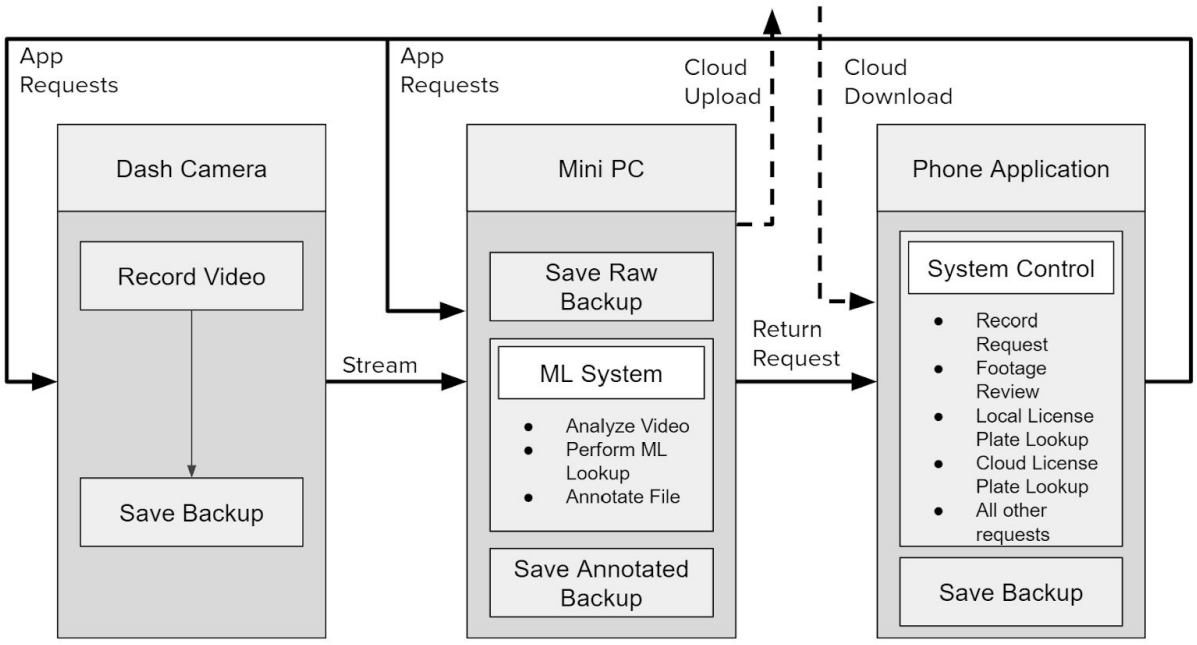


Figure 1.1: System Level Conceptual Sketch made by Evan Timmons

3. Statement of Work

3.1. Previous Work And Literature

OpenALPR - We're utilizing this service for license plate recognition because it has an acceptable pre-trained model on license plates.

3.2. Technology Considerations

Strengths

- Compute power on mini-pc's are adequate for our use case
- *OpenALPR* is accurate enough (99%) for our use case
- *Flutter* allows for speedy development with one single codebase

Weaknesses

- *OpenALPR* is not 100% accurate, so edge cases can be tricky
- Cellular service in rural areas is not sufficient for constant data uploading

Possible solution

- Create our own license plate recognition system that is more accurate
    - This will take a large amount of time with a lot of data required
- Only upload data while connected to a reliable source
    - This allows devices with data caps to upload only when connected to Wi-Fi

3.3. Task Decomposition

Hardware

- Hook external camera to *NVIDIA Jetson*
- Connect *NVIDIA Jetson* to *OpenALPR*
- Connect *NVIDIA Jetson* to mobile application
- Test runs

Software

- Develop *Flutter*-based mobile application
- Develop PostGRES database
- Connect mobile application with *NVIDIA Jetson*

3.4. Possible Risks And Risk Management

Distracted Driving

- Use of mobile application while driving can be distracting to the driver
- To manage, we are utilizing a simple interface to minimize distracted driving

Hardware Compatibility

- *NVIDIA Jetson's* built in camera is not compatible with *OpenALPR*
- To manage, we're hooking up an external camera instead

Privacy Concerns

- The data we are working with is personal and will stay personal
- To manage, we are going to have a clear and robust data management policy

3.5. Project Proposed Milestones and Evaluation Criteria

Key Milestones (in chronological order)

- Hardware Assembly completed
    - *NVIDIA Jetson* and external camera hooked up successfully
- Database Setup completed
    - *PostGRES* database with all necessary tables ready to go
- Mobile application completed
    - *Flutter* based mobile application with all requirements ready to go
- *OpenALPR* working with *NVIDIA Jetson* w/ camera
    - Able to successfully feed video data into *OpenALPR* and detect license plates
- Complete working product
    - Able to interface with recorded data inside mobile application

3.6. Project Tracking Procedures

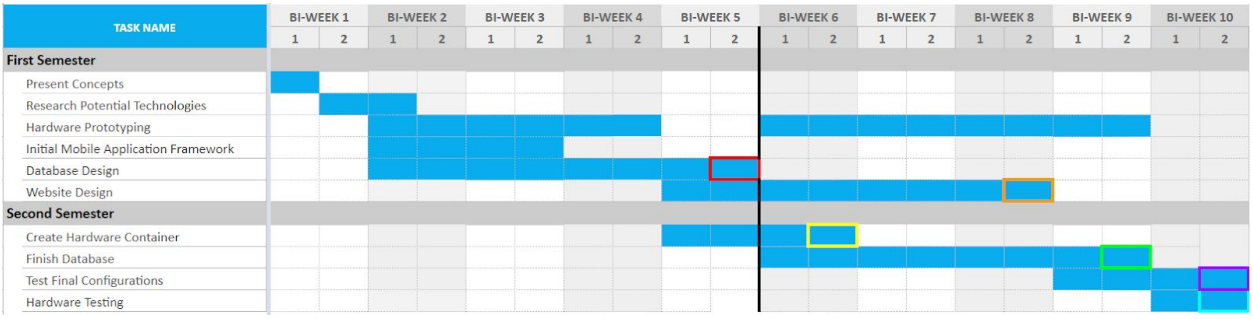
- *GitHub* for version control
- *Trello* for backlog and tasks to be done during bi-weekly sprints
- *Google Drive* for document sharing

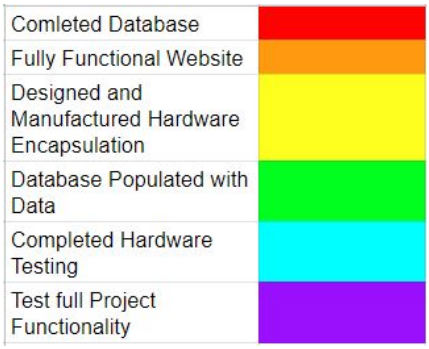
3.7. Expected Results and Validation

- Fully functional hardware consisting of *NVIDIA Jetson* w/ camera that interacts with our *Flutter* based mobile application and *OpenALPR* to successfully read license plates from a users car ride.
- Going to verify these expected results with real life examples.

4. Project Timeline, Estimated Resources, and Challenges

4.1. Project Timeline





Our Gantt chart for the project is shown above. In column one, the tasks are divided into two subsections, one for each semester. Tasks are mainly grouped by the semester of which they begin. We highlighted the key deliverables with unique colored outlines and explained them right below the chart.

Uniquely, our project has been hindered by the COVID19 pandemic. As a result, our tasks have been divided to account for the inability to host physical meetings. Notably, hardware prototyping was put on hold until our team is able to meet again next semester as we are unable to order new parts to continue the prototyping phase.

4.2. Feasibility Assessment

For a completed project, we would like to have the mobile application, web application and hardware completely developed and tested. Due to a variety of factors, mainly time and access to hardware, this is not likely feasible. Our hope is that we are able to complete enough of each part of the project to have a working prototype with all core features. However, it is likely that not every component will be completed.

4.3. Personnel Effort Requirements

Tasks to be Completed	Estimated Time to Complete
Present Concepts	The presentation of concepts only took our team 1 week to complete and was done during the first bi-weekly period
Research Potential Technologies	Researching technologies took our team a week to complete and another week to order the hardware components we had decided on.
Hardware Prototyping	Hardware prototyping is estimated to take 14 weeks to complete over two semesters.
Initial Mobile Application Framework	Initial mobile application framework took 4

	weeks to complete and allowed our software team to get a better feel for Flutter.
Database Design	Database design began our second biweek and will continue for 8 weeks until biweek 5.
Website Design	Website design is estimated to take 8 weeks over two semesters.
Create Hardware Container	Our hardware container will take us approximately 4 weeks to do and will account for stress testing of the finished product.
Test Final Configurations	Testing our final configurations is estimated to take around 8 weeks during only the final semester.
Finish Database Design	Finishing the database will take us 8 weeks to complete, beginning semester 2, and will take into account memory management and speed of getting information.
Hardware Testing	Hardware testing is estimated to take 2 weeks only during the final semester.

4.4. Other Resource Requirements

Other resources include, a car, a driver, a 3D printer, a usb webcam, and access to OpenAlpr license plate reading api.

4.5. Financial Requirements

There are many different hardware components and peripheral devices that are required for our project to be successful. Using the ETG to order these parts, we have come to the conclusion that the total financial resources required for our project is $450. This includes parts such as the NVIDIA Jetson TX2, a 1080p web camera, a SD storage card, and display monitor.

5. Testing and Implementation

5.1. Interface Specifications

We are going to use black box testing on our system if some part is not complete yet

5.2. Hardware and software

We will decide to choose the best testing hardware and software once we have some part done.

5.3. Functional Testing

We are going to test all the functionalities for our mobile application and website

5.4. Non-Functional Testing

We are going to test and make sure it supports a certain number of users at the same time

5.5. Process

We are going to record our tests as much as possible.

5.6. Results

Not applicable yet

6. Closing Material

6.1. Conclusion

In order to help drivers on the road, we need a better way to hold reckless drivers accountable. With our app and dash cam, we can do this. Furthermore, we can help insurance companies and the police get a better idea of people on the road. By meeting our specifications through the milestones we've set, we will be able to finish this project in a timely fashion.

6.2. References

OpenALPR software - Will implement in IEEE style in coming weeks

6.3. Appendices

Not Applicable Yet

List of Figures

Only figures used so far, made by Evan Timmons. Used in section 2.4 "Conceptual Sketch"
Additional figures will be cited here when used