**Method of Work**

The USU Parking and Transportation App is designed to help provide students with an easy way to access information to ease the stress of starting at a new university. The different components of the app use functionality from different systems in order to bring that information to one place. How those systems are integrated, alternatives, benefits they provide, and challenges are detailed in the section.

**USU Banner Integration**

The USU Parking and Transportation App takes advantage of several of the *Application Program Interfaces (APIs)* that USU provides. These APIs gather information from USU’s servers and report back the requested USU or Banner information to the app. The APIs included in the app are as follows:

* Student account ID
* USU account messages
* Student schedule
* Classroom buildings
* Instructors
* Live bus routes

The app administrator is given permission along with a set of credentials in order to connect to each of the APIs. Each time a request is made to retrieve information these credentials must be included and verified in the request. Once the request is verified the server responds with the correct information, which the app then displays on the appropriate page [].

The student schedule request for example is sent via a GET request, which is a method of *Hypertext Transport Protocol (HTTP)* used for communicating between clients and servers. The credentials given, or API key, is sent along with the GET request. The server then verifies that the key is valid and responds by sending the student’s schedule in either *JavaScript Object Notation (JSON)* or *Extensible Markup Language (XML)* format. The information from the JSON or XML object is then parsed out by the app. It is then formatted to be viewed by the user, or in this case it is used to form a route from the user’s current location to the classes reported in the returned object.

**Class Map**

The class map utilizes Google’s map API for most of its functions. Similar to USU’s API, the app will require an API key in order to send requests to Google’s servers. The key will be added into the app’s program code so it can be given when a request is sent. Google’s API has many features which are not pertinent to this project, so restrictions will be placed on the API key to ensure only authorized requests can be made. Restrictions are made in Google’s Cloud Platform Console, which is the same area where an API key can be requested [].

The map itself is added to the application with a Fragment object, which is simply an element to attach the map element. Once the map element is placed it will automatically handle operations such as: connecting to the Google Maps service, downloading map tiles, displaying tiles on the device, displaying various controls such as pan and zoom, and responding to pan and zoom gestures my moving the map and zooming in or out [].

The app then uses the schedule and course building information provided by the USU API to apply routes to each class. Paths can be placed on the map by placing coordinates, which are then connected by the map. This is accomplished by creating a poly line object and feeding it the coordinates in the form of latitudes and longitudes []. The specified points will be plotted out before hand over intersections of USU’s walking paths for the shortest path to be calculated.

The shortest path is found by making a map of possible nodes that can be traveled to from one point to another. This is accomplished by assigning a cost or a weight to each segment between nodes based on walking time. The shortest path is then chosen by finding the combination of paths to nodes between the start and end points. The weight of each segment will also be calculated based on how populated a certain path is. For example, a longer path might be given a lower weight if the traffic is light enough to observe a faster travel time.

**Parking**

The parking portion of the app uses cameras and AI coupled with machine vision, this is used to mark if a given parking spot is occupied or empty. This is a much cheaper and more easily implemented solution for detecting open spaces than other methods. A camera attached to a Raspberry Pi is placed facing towards an area of a parking space, and the AI determines if a n area of the image contains a car or not.

This is achieved by using Canny edge detection, which is a method where a filter, such as a Gaussian filter, is applied to an image to detect edges on objects. This turns an image into a black and white image with high contrast, which can be easily detected by a computer. The edges detected can then be determined to be certain shapes, such as a car, much more easily.

Certain areas of an image can then be marked as parking or driving areas, which will help eliminate false positive errors. The parking spaces will be marked as being empty or filled in real time and sent from each Raspberry Pi to the application. The parking information will then be updated on the map allowing the user to quickly see which spots are available.

**Alternatives**

One other option for displaying a map of campus was to simply create and integrate a map from scratch. This would allow us to provide our own proprietary implementation without having to rely on an external API. The downsides to this are cost of work, maintenance, and less versatility. Google’s Map API requires very little work to get it up and running, which cuts down on development cost and time. Once it is implemented it also requires less maintenance, as Google maintains their own API. Another reason to choose Google’s API is it allows the map to be used of campus for both normal map functions and creating paths to the student’s classes.

An alternative considered for the parking portion of the app was to install a sensor in each stall to determine if it was occupied. Several different methods were explored including pressure sensors, microwave radar, and ultrasonic sensors. Pressure sensors were initially selected over microwave radar and ultrasonic sensors due to its relatively simple design. The problem with all three solutions which ultimately lead to the current selection of machine vision was cost. The rejected solutions are simply too costly to implement and maintain. The current solution requires only a single Raspberry Pi and camera for a defined area and can be easily and cheaply replaced if damaged.

**Benefits**

The USU Parking and Transportation App will help provide valuable information to new students, which will help alleviate the stress of moving to a new area. The system will achieve this by combining several resources into one area to reduce the need to search for what is needed. Each student will have one central area they know they can go to answer many different questions and problems they may have.

**Challenges**

One challenge presented by the class map is keeping the shortest path algorithm up to date so it can be as efficient as possible. The weights assigned to each segment of a path will need to be updated depending on the current foot traffic. Depending on student class schedules, how many students use the app, and possible construction or other blockages will need to be considered as these metrics change. The paths will need to be updated each semester as well as updating individual path segments to account for impediments.