# Walk the Hills

*College Hill Association and Pullman 2040- Mobile Application*

**College Hill Association**

and

**Pullman 2040**

**Finite Cipher**

Adam Wagener;

Mitchell Footer;

Juan Ibarra-Delgado;

## TABLE OF CONTENTS

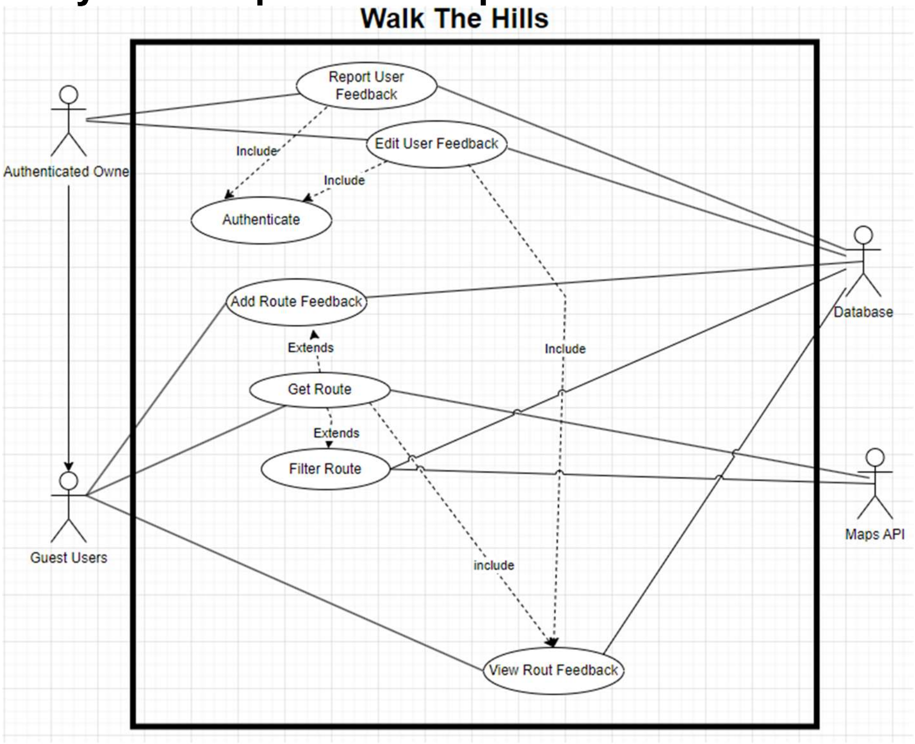
# I. Introduction

The College Hill Association and the city of Pullman recognized the need to enhance pedestrian safety and accessibility within the Pullman community. The growing concern for safe and efficient foot traffic in the area prompted these organizations to seek a solution that would empower residents and visitors to navigate the city on foot with ease. Traditional modes of transportation often proved inadequate or unsuitable pedestrian travel within the city. In response to this, the College Hill Association and the Pullman 2040 team turned to us to develop a practical, user-friendly application that could enhance and protect pedestrian travel in Pullman.

The motivation behind this project stems from a genuine desire to foster a safer, more convenient, and enjoyable environment for all people in Pullman be they residents, students, visitors, tourists, or anyone else. By providing a place to provide feedback on the local community as well as navigate from one place to another with an information source stocked with local insight, we strive to make walking in Pullman safer and better for everyone. Our vision aligns with the broader goal of creating a cohesive and vibrant community where people can explore and engage with their surroundings on foot as well as inform the city of any safety or maintenance ease with a simple and user-friendly interface.

The core objective of this project is to design and implement a robust mobile application that enables all people in Pullman to effortlessly find the safest routes from point A to point B. This application will take into account various factors, such as the time of day and user evaluated safety metrics, to provide real-time recommendations for navigating the city securely. By offering a solution tailored to the specific needs of the Pullman community, we hope to enhance the overall quality of life for its residents and foster a culture of pedestrian-friendly urban planning.

# II. System Requirements Specification



This application serves two main services that are handled through the functions as shown in the above UML diagram. The first of these functions we will discuss is providing walking routes for general users. These users should be able to access the application through a mobile device and gain a quick access view of walking instructions to get from point A to point B. This process should be accessible for all general users without needing log in or verification. In addition to providing a walking route, it is essential that the application also give users the ability to filter their route using multiple factors such as safety metrics, elevation, or fastest travel time. The get route function will be the bulk of the project as it services our largest stakeholder group.

In addition to routing and filtering, the get route function must also allow users to input feedback on the route they have taken. This feedback will be stored in a database to be accessed later. The route given will come from another stakeholder, Google, who provides us with their maps API.

The database object will provide a great deal of handling for route filtering, but also serves as a critical infrastructure to accomplish the second major task of the project. Reporting user feedback to city council members and the local police in an effort to get the city of Pullman the maintenance and safety information it needs to keep its population safe and happy. Through the database a registers council member or other authenticated user can access and edit the user input feedback to adjust route filtering. Accessing this data will allow them to make city changes that can then be used as justification for removing old database items such as streetlamp outages once fixed.

## II.1. Use Cases

**Get Route**

| | |
|---|---|
| Pre-Condition | Application launched |
| Post-Condition | Route information displayed.<br>Filter dropdown available. |
| Basic Path | 1. Locate location and destination fields.<br>2. Input destination.<br>3. input location field.<br>4. Select route.<br>5. Walk route. |
| Alternate Path | - In step 4 the user may select to add filters to their route and adjust their route before beginning to walk. |
| Related Requirements | - Maps API Integration<br>- Filter Route |

**Filter Route**

| | |
|---|---|
| Pre-Condition | Rout request received |
| Post-Condition | New Route returned, filtered based on selection |
| Basic Path | 1. Select filter dropdown and desired filter.<br>2. Select route.<br>3. View new route. |
| Alternate Path | - NA |

| | |
|---|---|
| Related Requirements | - Maps API Integration<br>- Database integration |

**Add Rout Feedback**

| | |
|---|---|
| Pre-Condition | Route in progress or completed |
| Post-Condition | Feedback stored in database |
| Basic Path | 1. Prompt for user feedback<br>2. Input feedback for safety rating<br>3. input feedback for other issues<br>4. submit feedback |
| Alternate Path | - Before step 1 the user may add a feedback pin to current location and input their own information to follow steps 3 and 4. |
| Related Requirements | - Maps API Integration<br>- Get Route<br>- Database Integration |

**Report User Feedback**

| | |
|---|---|
| Pre-Condition | Registered Owner Log In |
| Post-Condition | Route feedback data copied |
| Basic Path | 1. Registered Owner Login Authentication<br>2. Select feedback report.<br>3. Download file of feedback report from database |
| Alternate Path | - NA |
| Related Requirements | - Database integration<br>- Add route Feedback |

**Edit User Feedback**

| | |
|---|---|
| Pre-Condition | Registered Owner Log In |
| Post-Condition | Route feedback database altered. |
| Basic Path | 1. Registered Owner Login Authentication<br>2. Select edit feedback.<br>3. Select feedback pin to edit.<br>4. Adjust safety rating or other information report and change values.<br>5. Select save.<br>6. Select new pin or exit. |
| Alternate Path | - NA |
| Related Requirements | - Database integration<br>- Add Route Feedback<br>- View Route Feedback |

**View Route Feedback**

| | |
|---|---|
| Pre-Condition | Application Launched |
| Post-Condition | Feedback viewable |
| Basic Path | 1. Select view feedback.<br>2. Select feedback pin location. |

| | |
|---|---|
| | 3. Read displayed feedback.<br>4. Select new pin or exit. |
| Alternate Path | - NA |
| Related Requirements | - Database integration<br>- Add route Feedback |

## II.2. Functional Requirements

### II.2.1. Cross Platform Mobile Application

**Cross Platform**

| | |
|---|---|
| Description | The application must work on a number of mobile devices to be accessible to the general public. This requirement leads to the decision to work in JavaScript using the React framework. This code can then be packaged and run across an exceptionally large number of devices without major requirements by system. |
| Source | Internal requirement among team members to address College Hill Association desire for accessibility. |
| Priority | Priority Level 0: Essential and required functionality |

### II.2.2. High Performance Routing Accounting for Local Incidences

**Maps API**

| | |
|---|---|
| Description | The application must adapt to the ever-changing landscape of the city and provide accurate real-time mapping. This requirement leads to the need for Google Maps API integration. A mapping requirement of this size can only be handled by a team of hundreds with round the clock maintenance, so it is best to outsource this to a well-established reliable source such as Google. |
| Source | Internal requirement among team members. |
| Priority | Priority Level 0: Essential and required functionality |

### II.2.3 Dynamic Filtering and Citizen Voice

**Database Integration**

| | |
|---|---|
| Description | The application must allow users to provide input on what they are seeing. That input must be used to filter future routes. The need leads to the use of an internal database to store and filter user input safety ratings. |
| Source | City Council members on Pullman 2040 team asked for data input and retrieval. |
| Priority | Priority Level 1: Desirable functionality |

**Database Reporting**

| | |
|---|---|
| Description | The application must allow city council and the local police to view user input data so they can make adjustments to routes and city walkways. |
| Source | City Council members on Pullman 2040 team asked for data input and retrieval. |
| Priority | Priority Level 1: Desirable functionality |

**Database Editing**

| | |
|---|---|
| Description | The application should allow council members and other entities of authority to edit user feedback once issues have been addressed to keep the database and filters dynamic. |
| Source | City Council members on Pullman 2040 team asked for data input and retrieval. |
| Priority | Priority Level 2: Extra features or stretch goals |

### II.2.4 Safe and Accessible

**Get Route**

| | |
|---|---|
| Description | The application must provide pedestrian accessible public walking routes through the city of pullman with live location updates and adjustments for walkway closures. We will accomplish this through Google Maps API integration and our own database filters. |
| Source | Pullman 2040 desires a benefit to users in order to encourage feedback reporting. |
| Priority | Priority Level 0: Essential and required functionality |

**Route Filtering - Safety**

| | |
|---|---|
| Description | The application must allow users to alter their route to gain access to the safest route according to user input metrics. |
| Source | College Hill Association desires a safer walking application for their community. |
| Priority | Priority Level 1: Desirable functionality |

## II.3. Non-Functional Requirements

**Dynamic:**

The system shall update output routes and filtering using a dynamic adaptation. The system must respond to local closures and updates from city council and other users to continually adjust the overall safety and accessibility of routs.

**Collaborative Community:**

The system shall allow the local community to not only use the application, but also inform the local government of their needs and desires. The application shall report this information to better public transportation and public safety.

**Intuitive Use:**

The map instructions and user feedback inputs should be self explanatory and easily accessed by all members of the community regardless of technical skills or experience.

**High Performance:**

The system shall provide live walking instructions in real-time. Users should not need to wait to receive their next instruction while walking. The application will not delay travel in any way.

**Safe:**

The application shall provide safe routes where possible and will not route people down dangerous or impassible walking paths. The application will avoid private or protected paths.

**Private:**

The application shall not collect user data beyond that which the user provides such as location and user feedback. All other private information will not be collected unless provided voluntarily by the user.

## III. System Evolution

The most significant point of potential software degradation and maintenance comes from our use of the Google Maps API. This mapping software is brilliant, but it also comes with changing requirements and updates that may impact the life of our application. Google Maps API strives to be backwards compatible and provides support for old technology, but eventually updates may be required [1]. It will be the prime effort of our team to inform the technical board member in charge of software maintenance how to update the API integration when needed to ensure this process is painless and actionable. Though the Maps API will possibly cause need for software maintenance, it is still the only option moving forwards. It would not be possible to create an application as powerful as Google Maps as a three-person team in under a year with little to no funding [2]. The good news is that google maps and the react framework will continue to build cross platform designs that allow our application to transcend new generations of devices without major updates beyond the one mentioned above [1].

## IV. Glossary

**API:** Application Programming Interface. An interface provided by a third party to allow us to integrate their work and project into our own using our own custom functions built around theirs.

**Database:** A central hub of data to be collected and stored inside software.

**Data:** Any form of information that has been digitalized. Largely we are considering user feedback and user safety metrics when referring to data which will be stored in our database.

**Functional requirements**: The requirements for Finite Cipher that can be tied to a software functionality implemented by the team.

**Non-functional requirements:** The requirements for Finite Cipher that are general, often qualitative, and not related to a specific functionality.

**UML:** Unified Modeling Language. A standard format for creating diagrams to explain software processes and requirements.

## V. References

[1] Google Maps platform documentation | maps javascript API | google for developers, https://developers.google.com/maps/documentation/javascript (accessed Sep. 03, 2023).

[2] S. Koundinya, "Google maps vs. Waze: Which is the Best Navigation App," Guiding Tech, https://www.guidingtech.com/google-maps-vs-waze/#:~:text=Waze%3A%20Faster%20Routes%20and%20Traffic,on%20driving%20and%20riding%20directions (accessed Sep. 01, 2023).