CSCI P445
Software Requirements Specification

Cheyenne Pierpont
Bailey Forbes
Brandon Maurer

Individual Contributions

- **Cheyenne Pierpont**
    - I reached out to Cave Country Canoes & Alley Cat Advocates for sponsorship ideas for the project.
    - -I keep in touch with the owner of Cave Country Canoes on a sometimes weekly mostly bi weekly schedule on updates.
    - I have acquired the map that will be needed for the app on the river. I am also acquiring there logo
    - I have spoken to the owner and have the layout of the app of essentially what they are wanting it to do when the app is in use.
    - I write out meetings, notes, logs, discussions as well as I help in the state of setting up and writing out reports.
    - I set up the github repository link, and helped with adding in documents as we went along.
- **Bailey Forbes**
    - Contrived through the code structure of the project, specifically the back end.
        - Setting up and starting the Flask environment
        - Cryptology aspects of system such as ciphers and secure data distribution
    - Database structuring
- **Brandon Maurer**
    - Answered questions on RF1, set up file structure in GitHub repository

Key Personnel Information

- **Cheyenne Pierpont**
    - Personal Email: cheyennepierpont@outlook.com
    - University Email: cpierpon@iu.edu
- **Bailey Forbes**
    - Personal Email - prestonforbesbpf@gmail.com
    - University Email - bailforb@iu.edu
- **Brandon Maurer**
    - Personal Email - brandondmaurer@gmail.com
    - University Email - bramaur@iu.edu

Introduction

- Purpose:
    - Our project will be designed to help Cave Country Canoes improve the management of emergency distress situations and the handling of map-based data on their operations, among the critical tasks that are currently being done by them. With the emergence of such a modern system, the management of processes will be made more efficient, integrating the following:
        - a. Emergency incident tracking
        - b. Customer feedback management
        - c. Route and trip condition logging
        - d. Resource allocation for emergency response teams

- The system will be a web-based platform made with the help of Python's Flask framework, and the back-end data will be managed through a MySQL database. Primary features of the system are trip management functionalities, including logging customer trips, recording points of interest (POI) and geofences and sending emergency alerts. Also, to ease the management of the resources and response teams, sending processing of the same will occur, thus ensuring quick encapsulation and response during times of emergency. Data security and continuous updates are improved by the automatic triggers for logging changes within the accounts, customers, turns, and emergency incidents. This layout is designing a trustworthy, responsive system from both the customer and in-house side. By using the services of Heroku, the application will be online and accessible to both the staff and users. This solution is a scalable one, which means it can be used to manage both day-to-day and emergency operations.

- Definitions
  - Python's Flask Framework: Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier.
  - MySQL: is an open-source relational database management system. It is based on structured query language.
  - Heroku: is a cloud-based platform that helps developers build, run, and manage applications.
- System overview
  - This will be a mobile app that can be downloaded onto phones, also access for the company through desktops/phones. It will have tabs on it essentially under one tab there will be a map of the river. On this map there will be a button saying “Contact Representative from Cave Country Canoes with location for emergency”. The user will send their location from the map to CCC. Essentially this will be sent to who is working that night and the information can be sent to the right Emergency services and get help to them faster.
- References:
  - These are two apps that we are using for references on the app for the layout.
    - https://www.strava.com/
    - https://paddling.com/paddle/go-paddling-app

Overall description

- Product perspective
    - System Interfaces:
        - Flask Python Framework: Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier.
        - MySQL: is an open-source relational database management system. It is based on structured query language.
        - Heroku: is a cloud-based platform that helps developers build, run, and manage applications.
    - User Interfaces: Currently the plan for the user interface is to have the map to be the main screen and the user can tap where they are, and a little menu will pop up with a button allowing them to send that area as their location.
    - Hardware interfaces: - if a physical computer acts as a server. However, the current solution is cloud-based, so this section can describe how it interacts with user devices (phones, computers).
    - Software interfaces: Works with the use of MySQL, Flask, and cloud hosting via Heroku. This section should detail how these components interact with each other.
    - Communication Interfaces: - This section could detail the communication methods, such as the network interfaces between mobile apps and the server (Heroku)
    - Memory Constraints: This could be expanded with specific memory requirements based on the proposed functionality and technologies.
    - Operations: - The system’s operational requirements (e.g., 24/7 availability, emergency response systems, automated alerts).
    - Site Adaptation Requirements:
    - Product functions: primary functions: emergency incident tracking, customer feedback management, route logging, and resource allocation.
    - User characteristics: ease of use for both customers and staff. Further UI details would need to be added, such as mobile and web interface designs.
    - Constraints, assumptions and dependencies: constraints such as data security and compliance with various standards (FCC, FTC, IEEE). Assumptions include the availability of MySQL and Flask for building the system.

Specific requirements

- External interface requirements: External devices/services are emergency personnel systems and user devices.
- Functional requirements: Tracking, feedback management, and resource allocation.
- Performance requirements:
- Design constraints: Data security, privacy, and accessibility
- Standards Compliance: the system's expected performance (response times, maximum concurrent users, etc.) should be detailed here. Logical database requirement
- Software System attributes:
    - Reliability: The system's reliability can be described in terms of its redundancy (cloud-based) and error handling mechanisms.
    - Availability: Cloud hosting via Heroku, should allow high availability
    - Security: Data privacy, collection requirements, and compliance with security standards.
    - Maintainability: The choice of technologies (Flask, MySQL) allows for maintainability due to the widespread support for these technologies
    - Portability: Cloud hosting and mobile access, allowing it to run on various platforms.

Other requirements

- The system's reliability can be described in terms of its redundancy (cloud-based) and error handling mechanisms.