CSCI P445
Software Architecture Specification

Cheyenne Pierpont,
Bailey Forbes,
Brandon Maurer.

SAS

1. Proposed software architecture

1.1. Overview

The proposed system for Cave Country Canoes is a web-based emergency management platform. It will enable real-time emergency distress tracking and customer feedback management. The system will be built using Python's Flask framework for the backend and MySQL for persistent data management. It will be deployed on Heroku, providing scalability and cloud-based access for both staff and customers.

1.2. Subsystem decomposition – Identify the subsystems and the responsibilities of each. You should use component diagrams.

Emergency Management Module: Responsible for logging emergency incidents, and letting owners know where customers are on the river.

Customer Feedback Module: Allows customers to provide feedback on trips and app, which are stored and analyzed for service improvement.

1.3. Hardware/software mapping – How will subsystems be assigned to hardware? You should use deployment diagrams.

Server: Flask-based backend hosted on Heroku, managing all system logic and database interactions.

Database: MySQL hosted on a cloud-based service (Heroku's database), ensuring accessibility from multiple locations and devices.

Client-Side Devices: Mobile devices, enabling access to the system for customers and employees.

1.4. Persistent data management – Identify the data which will be persistent. Describe the file system or database to be used, including a complete database design.

Data to be Persisted:

Emergency Incident Logs: Incident reports, timestamps, locations, and response actions.

Feedback Data: Customer reviews and ratings after each trip.

Database Design:

needs to be added

1.5. Access control and security – For each different actor (user, system administrator, etc.) describe the operations they will be enabled to use. Describe authentication and security provisions.

Actors:

Customers: Provide feedback, and send distress calls.

Staff: Can access emergency reports, allocate resources, and manage customer feedback.

System Administrators: Full access to all system functionality, including database management, and log auditing.

Authentication and Security:

Role-based Access Control (RBAC): Enforce different permissions for customers, staff, and administrators.

User Authentication: Implement secure login mechanisms, such as OAuth for staff and customers.

Security Provisions: SSL/TLS for data encryption in transit, hashing of user passwords, and logging user activities for security monitoring.

1.6. Global software control – Describe the control flow (e.g. procedural, event-driven, threaded). Procedural control flows should be described using activity diagrams. Event-driven flows are best described using sequence and state diagrams (use UML diagram standards).

Control Flow:

Event-Driven Flow: When a customer sends an emergency alert, it triggers the emergency management system, notifying staff and logging the incident.

Procedural Control: Routine functions like user login, trip logging, and feedback submission follow a linear flow, involving form submission and database interactions.

1.7. Boundary conditions – describe how the system will be started up, initialized and shut down. How will it respond to errors and exceptions? Any daily/weekly/monthly/yearly efforts necessary? All organizations buy new computers every 3-4 years, how to migrate to a new server? Be able to bulk dump all data to file and bulk load all data from the same file.

Startup and Initialization:

System Startup: Deployed on Heroku, with database migrations to initialize MySQL.

Error Handling: Flask's error-handling framework will manage exceptions, while automated backup scripts ensure data recovery in the case of a server failure.

Daily/Weekly/Monthly Tasks: Regular backups of the MySQL database (e.g., daily using mysqldump). Routine server health checks and updates.

Server Migration: Every 3-4 years when new hardware is procured, data migration would be handled through bulk data export/import using tools like mysqldump for MySQL and reconfiguring the Heroku deployment.

- Regardless of the techniques being used, we can say that any system can be said to be composed of nine basic component types:

  - Use Cases
  - Functions
  - Triggers
  - Data Stores
  - Data Flows
  - Data Elements
  - Processors
  - Data Storage
  - Data Connections
  - Actors/External Entities

Use Cases are an ordered set of processes, initiated by a specific trigger (e.g., transaction, end of day), which accomplish a meaningful unit of work from the perspective of the user.

Functions are context independent processes that transform data and/or determine the state of entities.

Triggers are the events that initiate Use Cases. There are three types of triggers: time

triggers, state triggers and transaction triggers.

Data stores are data at rest. Data flows are data in movement between two processes, a process and a data store, etc.

Data elements are the atomic units within data flows and data stores.

Processors are the components which execute the processes and events (i.e., computers and people).

Data storage is the repository in which the data stores reside (e.g., disks, tapes, filing cabinets).

Data connections are the pipelines through which the data flows flow (e.g., communications network, the mail).

Actors/External entities are people or systems outside the scope of the system under investigation but with which it must interface.

Each of these components has many properties or attributes which are needed to fully describe them. For example, in describing a process we can state its algorithm, who or what executes it, where it takes place, when it takes place, how much information it must process, etc. Figure 1 lists the properties which can be described for the various component types.

In a given project and for a given component, the properties which must be gathered/defined may vary. The SDLC must allow for this flexibility versus an all-or nothing approach.

Individual Contributions

- CheyennePierpont
    - 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
- BaileyForbes
    - 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
    - Helped fill out RF2