Corvallis Flood Watch

Scott A Premo , Christopher Asbury , Price Howard , Connor G Mclaughlin
CS462
Project Partner - Kirsten Winters
Scope and Vision Assignment - Due 3/8/2024

Abstract

In this document we have aggregated our scope and vision assignments from Fall term 2023 into one document to highlight what parameters have changed in the last 3 months. Last fall, our project moved on from originally being a hardware-based project with no web app and a different project partner. We have added and changed these documents to adjust the vision and scope of the project to more accurately fit our project as it stands now. Everything that was changed or added is marked with a highlight, and possibly a strikethrough depending on the nature of the change.

Software Development Process (SDP)

Principles

- We are constantly checking for discord notifications, emails, and text messages from each other and our project partners so that we keep in touch.
- We are critical in our feedback, but also not rude or uncouth
- We must have at least 1 person review and confirm pull requests
- Everyone must have at least 1 task to be working on after each group meeting
- Everyone attends meetings on time, or has communicated a reason for missing the meeting at least 24 hours in advance, barring extenuating circumstances.

Process

- Originally our project was going to undergo a very waterfall-esque process.

- In its current form, our project will be taking on a much more iterative development approach. The key difference here being that we will be testing our project as each step progresses, rather than saving most of the real testing for the end of the project.

Roles

Our team does not have a definitive “leader” or “scrum master”. Everyone contributes ideas, and everyone keeps one another accountable for their actions.
Lead communicator - Connor

Tooling

Version Control	GitHub.
Project Management	GitHub Issues and Projects, Discord.
Documentation	README,.md
Test Framework	Jest, Postman, Insomnia
Linting and Formatting	Prettier, Pylint
CI/CD	GitHub Actions.
IDE	Visual Studio Code, VI.
Graphic Design	Figma, Pen and Paper, Google Slides.
Others	Various documentation websites (too many to list).

Definition of Done (DoD)

- Acceptance criteria are validated
- Changes are merged
- Tests are successful
- Changes are implemented
- Documentation is updated
- Changes are deployed to staging
- Demo is prepared for next meeting

Product Requirements Document (PRD)

Problem Description

Currently available solutions for road hazard notification often only notify a user about hazards on their current travel path, and rely on crowd-sourced data exclusively. There is a distinct lack of in-house data being applied to current solutions.

Scope

~~Our project's theoretical scope is limited to wherever we can get verified, accurate map data. Currently, we are limiting our scope to the Corvallis area, as we will not have the resources to expand as of right now.~~

~~We will be designing our app under the limitation of the actual scope, but with the theoretical scope in mind (modularization).~~
Our Project's scope is limited to wherever we have enough users, as well as wherever we can set up sensors for automatic data collection. Currently we are set up so that we can have the application function globally, but we are currently only going to keep it localized for English speaking countries. The sensor data is going to be limited to Corvallis Oregon.

This change came up from the fact that it was easier to get global map data than just the map data of Corvallis.

Use Cases

We want to open the app and use the "Hazard Selection" tool to filter which hazards we have on the map at once.

The user will be able to report observed hazards using the "Report Hazard" button.

The user will be able to view past data by being able to select dates in the hazard map.

Purpose and Vision (Background)

Our purpose is to develop a hazard watch platform that provides our users with current and past data that is selectable to what the user wishes to watch out for.

We want to become the platform of choice for people planning trips and wanting to avoid road hazards.

Up to now, you have always been at risk of having your path blocked from: random flood zones, construction blocking paths, landslides or car crashes.

Stakeholders

- Seed Investors
- CEO/CTO/Founders
- Engineering Managers
- Product Managers
- Engineering Team
- Marketing Team
- Legal Team
- Sales
- Users
- Local Government Municipalities

Preliminary Context

Assumptions

We can integrate map data easily and dont have to reinvent the wheel in that regard.

We have until the end of Winter Term to have our website at an adequate level of development to begin testing it with hardware instruments.

Constraints

We are a small team, and should plan accordingly. We want to be able to test core functionality of the website by the end of winter term

Due to our data being reliant on crowdsourcing for part of its data, our data is limited based on our user base.

Dependencies

We need a database to store the past and present data that is reported.

We are dependent on users uploading data for us to be able to process.

Hardware dependencies based on the sensors we integrate with the project.

Market Assessment and Competition Analysis

Alternatives:

- Apple Maps, Google Maps, Waze: Good alternatives but are not primarily made to report hazards or view them.
- Pulse Point: Police / emergency services reporting app which only gives notifications when emergency services are responding to it.
- Street bump: A crowdsourcing app that collects data about potholes while you drive to share with local governments.

Target Demographics (User Persona)

- Jenkins is a 47 year old city planner who is interested in fixing poor road conditions in their city, but lacks data of where these flash conditions occur.
- Sam is 24 years old commuter who has to drive to their work every day and cant afford to be late
- Leeroy is a 23 year old college student who is empathetic and likes to report issues whenever they can so that others can avoid them.

Requirements

User Stories and Features (Functional Requirements)

User Story	Feature	Priority	GitHub Issue	Dependencies
As a commuter I want to be able to view what sort of hazards exist in my town and know how it might affect my day.	Overhead Map	Must Have	TBD	Web App Feature working
As a student in a college town I want to report when I see something happen, so that my peers can be safe.	Report Hazard Feature	Must Have	TBD	Web App Feature working
As a city planner I want to be able to see past data regarding road hazards in my area By being able to view data in an analytical view.	Hazard filters/data	Must Have	TBD	Physical Sensor is connected to the database.

Non-Functional Requirements

- The website and database should be able to handle increases in traffic from users during peak periods
- The website should be available and formatted properly to be able to be used on various devices.
- The code would be well documented.
- The website should be user friendly and easy to navigate.

Data Requirements.

Integration Requirements

Unknown at this time, but we will probably use a map api of some sort.
We will update this document upon any api usage.

User Interaction and Design

Provide wireframes, mockups, or visual representations of the product's user interface. Describe the user experience design principles and interactions. Can live in a dedicated issue or document.

We have not had time to sit down and design any UX yet, but we have a general idea of what some features will look like.

- We want the overhead map to have a checkbox feature that allows users to filter what kinds of hazards show up on their map
- We want the Report feature to have all of these options available to report
- We want the flood data to be able to show an average and a current datapoint for a given sensor that can be clicked on on the map
    - Possibly the feature to look at all sensor data in a list as well

Milestones and Timeline

Website-only functionality done by the end of Winter Term 2024

Project done by the end of Spring Term 2024

Goals and Success Metrics

Goal	Metric	Baseline	Target	Tracking method
Verify Accuracy of data	Average of sensor input	Currently Unknown	>95%	Checking sensor data from the physical sensor and cross-referencing to see it is being uploaded correctly.
Product-market fit	How would you feel if you could no longer use this product?	Currently Unknown	Very disappointed > 40%	Interview

Open Questions

What will the physical hardware even be?

What does it look like to have the data from this hardware uploaded automatically?

Will it still need to be battery powered?

Out of Scope

The project will not include any physical infrastructure beyond the implementation of the physical sensors. For example, while our project concerns itself with informing municipalities about things such as flood data, it would not be responsible for actually building any of these infrastructure improvements.

Software Design and Architecture

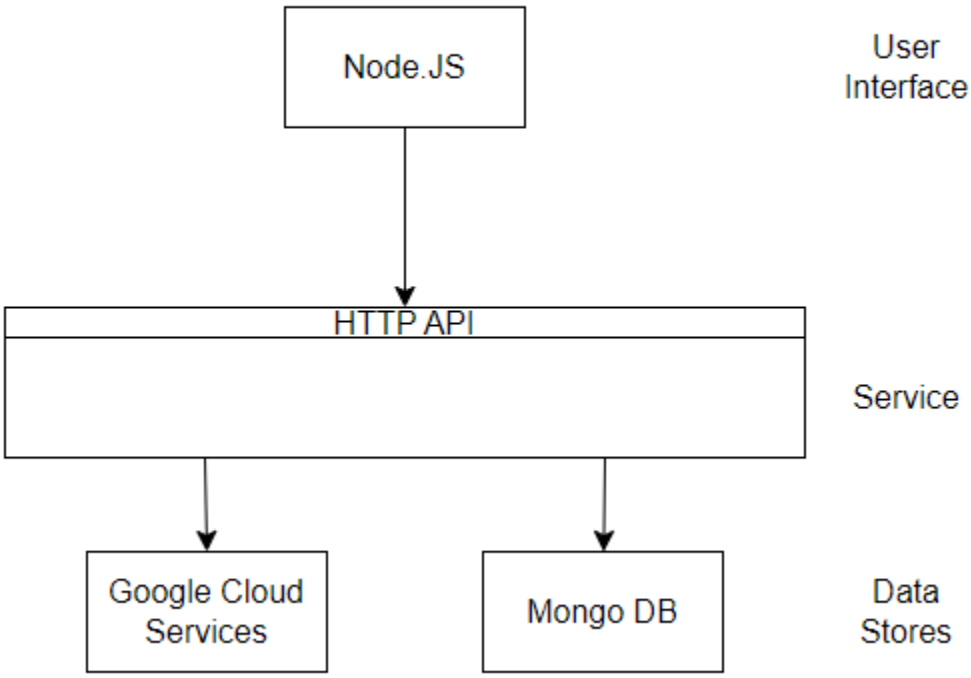
Introduction

This document outlines the architecture for a web-based road monitoring system. A well-defined architecture ensures scalability, maintainability, and reliability, it aligns with our goals as a group and the goals of the class.

Architectural Goals and Principles

The key objectives our architecture aims to achieve are multi-platform access, scalability, and security using the principles of separation of concerns, encapsulation, and modularity.

System Overview



CHANGES: Our system overview has changed a small amount. While we're still using node.js we're also using React to implement the front end of our website. We are also using Supabase instead of Google Cloud Services and a SQL based database instead of MongoDB. These changes are due to better functionality. Adding React allows the front end to be handled more efficiently and the code is organized better. We also found Google Cloud Services was not a good fit for hosting the database and found Supabase was a better alternative that could connect better.

Architectural Patterns

We have chosen to use the microservices design architectural pattern. We chose this architecture as our project will have several small services that will interact with each other to be able to create our finished product.

Component Descriptions

- User Interface: Handles user interactions with our web service/application, displays map data and data associated with flooding or other hazards, provides support for finding data in the local area and viewing common hazards.
- Backend Server: Process API calls, send responses and handle inputs, Interface with the database to enter data and retrieve data.
- Database: Stores and manages data using a non-relational database system.

We changed to using a relational SQL based database as it fits our database better as we do not have a very complicated database.

Data Management

Our data will be structured using a non-relational database scheme for hazard data, user profiles, and location data. The data will be accessed using RESTful API endpoints for CRUD operations on the hazard data, location data, and user profiles.

Same as above switched to a relational SQL based Database.

Interface Definitions

Hazard related API Calls
GET /hazards will return a list of all reported hazards in the database
GET /hazards/location returns the list of hazards in a specified location that the user is viewing on the map
POST /hazards/location will create a new hazard report in the database
GET /hazards/<hazard_id> can show a specific hazard's data
DELETE /hazards/<hazard_id> will delete a specific hazard, will only be accessed by admin profiles
*GET /locations returns a list of all locations, along with positional map data for that location. (for creating the map overview)

User Profile API Calls
GET /users/<user_id> Retrieve user information, accessed by the user themself or admin profiles
POST /users Create a new user profile

- We are no longer creating users in the same sense as before, we are now just using Auth0 and we will put their username related to a user_id. We are no longer storing passwords.

PUT /users/<user_id> Update a users profile, accessed by the user themself or admin profiles

- To clarify, there will be no updating user information other than reputation

DELETE /users/<user_id> Delete a users profiles, accessed by the user themself or admin profiles

*GET /users gets all current user profile names.

*indicates an endpoint that may not make it to the final stage of development, depending on group decisions.

- We don't have this endpoint in our plans anymore

Considerations

Security

One security risk is being able to spam wrong information to our system, which will make our data irrelevant. One method to be able to fix it is to monitor rates that which accounts submit data, or rates that IP's submit data and if too much is submitted in too small of a time frame we can black list that account/ip

If we have user data, a security risk is protecting that data. A way to fix this is by running a strong encryption algorithm to protect the user information.
Instead of an encryption algorithm we are just using Auth0.

Performance

Discuss performance requirements and strategies to meet them, i.e., scalability, load balancing, and caching mechanisms.Some examples: horizontal scaling using container orchestration, local storage for the web application, etc.

Maintenance and Support

Once 1.0 is complete this project will be handed to OSU and they will handle any updates that they decide are necessary for the project. It is at this point that Oregon State can determine whether they would like to continue the project with a new group to potentially be developed further, if the project will remain as-is with some dedicated maintenance, or if the project will be terminated upon our departure from the university.
We still plan on handing this project off to OSU or an industry client, but it will be beyond its 1.0 stage most likely.

Deployment Strategy

Our architecture will be deployed to either git pages or through google cloud hosting(undecided currently). We will have our main branch be hosted, but will reserve private branches for testing purposes that will be inaccessible to the public.
We are walking back our idea to deploy on google cloud, and are still researching a hosting platform.

Testing Strategy

We will primarily use applications like Jest, Postman, and Insomnia to test the API calls, database integration and user interface.

Glossary

API: Application Programming Interface
Crud: Create, Read, Update, Delete.