CS Capstone Progress Report

December 3, 2018

HyperRail App

Prepared for

OPENS Lab

Chet Udell

Prepared by

Group 25

Vincent Nguyen
Adam Ruark
Yihong Liu

Abstract

This document discusses the progress and current standings on our work in developing the HyperRail application. It also describes the direction this project will take and some ideas for the future.

Contents

1 OVERVIEW

The HyperRail system is a set of sensors attached to a robot that travels up and down a rail. While the robot travels, the sensors periodically take measurements, such as temperature or humidity, and record them. The hardware for this system has already been chosen and implemented, leaving us with the This purpose of this project is to create a web-based application to improve the usability of this system, enhance current features, and to add new features. Some goals of this application are to allow users to save configurations for future runs, schedule runs of the HyperRail system, and to provide more clarity on the status of the HyperRail while it is running.

2 RETROSPECTIVE

Over the last ten weeks we've performed a significant amount of research into what needs to be done to complete this application. Below is a table of weekly events and findings.

Positives	**Deltas**	**Actions**
Resumes were updated and long-term career goals were discussed.	Improve the content precision and styling of our resumes.	Received feedback from other students to help improve our resumes.
We decided on 10 projects that we were willing to work on.	None	None
We met with our client for the first time and talked about the project overview and goals.	Think about how to implement the HyperRail Application.	Started researching web development and database languages and tools, and analyzed the existing code.
Problem statement is finalized.	Figure out how to solve the problem now that the problem is finalized.	Kept looking into possible solutions for the project.
We decided on requirements for the entire project.	Think of technologies that can fulfill the project requirements.	Started researching and deciding on technologies to use for the project.
We decided on team standards and compiled our research and decided on various technologies that we could use for the project.	Learn how to use the selected tools to implement the HyperRail App.	Started experimenting and continuing research with the selected tools.
We finalized our tech reviews, incorporating the feedback received from other students.	None	None
Started detailing our implementation plans for the design document.	None	None
Finished up our write ups and continued with research and experimentation.	None	None
Reviewed our progress throughout the term.	Start looking at user interface (UI) designs for the web application.	Started making UI diagrams.

3 Current Progress

The end result of this term's efforts have been a decisive choice in how we want to execute our plan. The over-arching structure of this project will consist of three components: the user interface, the central server, and the HyperRail hardware. In general, the hardware doesn't care how the user interface is implemented and vice versa. The progress on this application has been in research and testing of possible options for us with little to no actual implementation.

For the server and database components of this application, Adam Ruark has been experimenting with a NodeJS controlled server with a MongoDB database on the back-end. Below is some inital code that he has created while testing this configuration.

```
const bcrypt = require('bcrypt');
const express = require('express');
const app    = express();
const port   = process.env.PORT || 3000;
const mongo = require('mongodb').MongoClient

let db, collection;

// Requests choose their handler in the order they are defined
app.use(express.json());
app.use(express.static('public'));

/* Expects:
   {
      username: <username>,
      password: <password>
   }
*/
app.post('/createUser', async (req, res) => {
   const username = req.body.username;
   const password = req.body.password;
   if(username == null || password == null) {
      res.status(400).send('Invalid payload');
      return;
   }

   const exists = await searchDatabase(username);
   if(!exists) {
      const hash = await bcrypt.hash(password, 10);
      await createUser(username, hash);
      res.status(201).send('Created');
   } else {
      res.status(400).send('User Already exists');
   }
});
```

```
app.get('/status', (req, res) => {
   res.send(`Server is good`);
});

app.get('/listUser', async (req, res)=> {
   let query = {};
   if(req.query.user != null){
      query = {username: req.query.user};
   }
   const data = await collection.find(query).toArray();
   res.json(data);
});

// Start database connection
mongo.connect('mongodb://localhost:27017/hyperrail', {useNewUrlParser: true})
   .then((client) => {
      db = client.db('users');
      collection = db.collection('users')

      // Start API server
      app.listen(port, () => {
         console.log(`Server started on port ${port}`);
      });
   })
   .catch((err) => console.error(err));

/******************
 * Helper functions
 ******************/
// Check if user currently exists in database
async function searchDatabase(username) {
   const query = {username: username}
   const res = await collection.findOne(query);
   if(res == null) {
      return false;
   }
   return true;
}

// Add user to database
async function createUser(username, password) {
   const obj = {username: username, password: password};
   const res = await collection.insertOne(obj);
}
```

The code above initializes a server on port 3000 and a local MongoDB instance for use by the server. Users can hit

a couple of proof-of-concept endpoints and this information gets processes and uploaded to the database. Most of this code will be rewritten over the course of the next term as the application progresses. The next steps for this piece of the project are to develop an API for the user interface and also work out how this portion will fit with the hardware we have. The latter task requires some discussion with potential users and our client to discover the restrictions that should be considered.

Vincent Nguyen has been looking into client-side development by looking into web development languages, such as HTML, CSS, JavaScript, user interface design, as well as various APIs that could help with styling or optimization, including jQuery and Bootstrap. Before the group decided to use the MongoDB to host the data collected, Vincent has also looked into PHP and database querying using asynchronous AJAX requests. There has been no experimental web page and no UI decisions have been officially made either.

Yihong Liu has been looking for the prototype design information that requires, such as user storyboard, data collection. And Yihong has started to design an prototype in farmer.js with the client's requirements, but there is still something that we all need wait until we have a meeting with our client to talk about the design details, but the most layout and design style is confirmed, besides that, Yihong has been looking for HTML, CSS, Javascript with different IDES that we are going to use, and some possible ways to test our GUI after we have done the UI code. the draft layout is here:

- Account: Provides views for account management, including login, registration, password recovery, login failure, email confirmation, other related account information.
- Facility: Provides views for facility staff to manage residents of their facility who own the Hyperrail system, and some related information about term of service.
- Home: A home page with general information about the Hyperrail system, such as documentation and contact information on the top bar.
- Setting: Provides additional views for account management beyond login, including password changes, binding other information to the account, logout, etc.
- Project: A place to save an unfinished research or create a new research, for example, the user can get the research process and the data from last time they use the Hyperrail system. Also, the user can create a search with different command parameters.
- Operating page : A page shows the correspond information that request the user to input, such as ”rail length” , ”spool radius” , ”velocity” , ”whatever user wants to test”, and user can change the input.
- Data information : the page shows the collected data from the sensor on the hyperrail system after the user entered the command parameters.

And Yihong will do the improvement after the client and us reach a consensus on the details.

4 PROBLEMS

The only issue that the team has encountered this term is not having a physical HyperRail setup to experiment with until recently. Because of this, we have been unable to experiment with the hardware interactions with the rail. While the OPEnS Lab staff have been constructing an experimental setup, the team has been researching or experimenting with web application development.