Milestone 5 Report

Engicoders
H. Bird, B. Karacelik, J. Peters, J. Ropotar, A. Rybka
Department of Computer Science, University of British Columbia
COSC310: Software Engineering
Dr. Gema Rodriguez-Perez
April 12th, 2024

- **(2%) Update the requirement document, including:**
    - **A brief description of the software you are building**
    - **A list of user groups for your software, along with an example scenario for each user group of how they will use the software**
    - **The final list of requirements of the software that was built. Incorporate feedback from M2.**

Our team has developed a Manufacturing Operations Webpage aimed at enhancing management's ability to forecast line scheduling by leveraging Machine Learning predictions based on existing data. As an Operations Manager, gauging the demand for specific product customizations can be challenging. Take Toyota, for instance, which offers various trims, colours, and specification options for its customers. When optimizing a manufacturing assembly line, it's crucial to anticipate where demand will lie for particular features, enabling the creation of appropriate schedules and meeting deadlines. As a group of Manufacturing Engineering students, we recognize this challenge firsthand and sought to create a software application that empowers users, such as Operations Managers, to utilize existing datasets and machine learning algorithms for more accurate demand prediction.

The project gathers IoT data via an API from the "Thinkspeak" IoT cloud platform, covering various data types like pressure, temperature, and speed, as long as they're time-stamped. Targeting manufacturing engineers, the software includes data cleansing with filters and outlier removal. Sk-learn ML algorithms forecast future trends. Alarms alert users to high/low sensor values. Processed data is visualized for insights and exportability. An alert system identifies critical sensor values.

The software offers various functionalities tailored to regular users' needs. Users can test the software for free by selecting a preset database from a drop-down menu. They can filter and cleanse data, ensuring accurate visualizations. Users can choose from different visualization types and personalize them by manipulating the axes. Additionally, they can plot the direction of their data to predict future results and visualize multiple sensors simultaneously for comparison. Setting alerts for dangerous sensor values and managing alarms to prevent common issues like chattering are also available. To ensure confidence in data-driven decisions, users can view the range of accuracy for predicted data. Finally, users can download visualizations and alarm logs for future reference, facilitating easy access to saved data and analyses.

Status of Software Implementation

<u>Implemented Requirements</u>

W

- Sign up

- Sign in
- Upgrade
- Select data from api and use
- Filter data
- Select visualization type
- High low limits
- Alarm mngmt
- Prediction
- Display results
- Export results
- Alarm log
- Export alarm log
- Signout
- Premium upload

- Seamless selection from API
- The webpage should run on any modern device, including mobile.
- The prediction algorithm should not take more than 1 minute to process future data
- User input data must be securely stored, as it could include confidential information
- The webpage must be able to handle multiple user interactions at the same time

Unimplemented Requirements

Initially, our project intended to allow users to input or select multiple sensors to visualize in the end. Unfortunately, we could not implement this into the program due to time constraints as well as it would cause difficulties with the way our project takes inputs from users. We also intended to have preset data from an API in a list for users to easily select, but now we make the users get the channel and route id from a public channel in thinkspeak to access data to use in the program. Finally, the users are not able to manipulate the axis of the charts since that can not be implemented with the selected chart import. Chart.js automatically chooses the best axis to view the input data anyways.

Backlogged Tasks

When problems arose with user stories and tasks that were either changed, or not possible, the tasks were altered and implemented or marked as not feasible and closed respectively. This left no tasks in the backlog by the end of the project, but the notable unfinished or changed tasks are: task #34 multiple sensors which was marked as unfeasible, task #109 and #132 which changes the preset data list to inputs from the user which gets data from thinkspeak, and task #259 axis manipulation which was marked as unfeasible and unnecessary due to chart.js doing the work anyways.

System Architecture

W

Code Reuse

W

Known Bugs with Solution

w

- How many of your initial requirements that your team set out to deliver did you actually deliver (a checklist/table would help to summarize)? Were you able to deliver everything or are there things missing? Did your initial requirements sufficiently capture the details needed for the project?
- If you have any requirements unimplemented, you will want to include an extra section showing those requirements have not been implemented.
- If you have a requirement that is partially working but feel that there's enough you don't want to call the whole requirement as incomplete, then break it up into two requirements, with one shown as tested and working, while the other is shown as not working.
- How many tasks are left in the backlog?
- What is the architecture of the system? What are the key components? What design patterns did you use in the implementation? This needs to be sufficiently detailed so that the reader will have an understanding of what was built and what components were used/created. You will need to reflect on what you planned to build vs what you built.
- What degree and level of re-use was the team able to achieve and why?
- Indicate any known bugs in the software.
- Explain how these bugs can be fixed.

Project Handover

W

- Installation details needed to deploy the project
- Dependencies needed to deploy the project
- Any maintenance issues required to run the project (e.g., account information)

To run this project, there are 2 main steps. Download the repository from github onto a machine with Python 3 installed. Run the command pip install -r requirements.txt. This will install all the requirements. From there, run py -m flask

run, and the app will run on the localhost, 127.0.0.1. It is recommended to run the test cases before running the main app, as some database

Reflections

W

1. How did your project management work for the team? What was the hardest thing, and what would you do the same/differently the next time you plan to complete a project like this?
2. Do you feel that your initial requirements were sufficiently detailed for this project? Which requirements did you miss or overlook?
3. What did you miss in your initial planning for the project (beyond just the requirements)?
4. Would you (as a team) deal with testing differently in the future?
5. If you were to estimate the efforts required for this project again, what would you consider? (Really I am asking the team to reflect on the difference between what you thought it would take to complete the project vs what it actually took to deliver it).
6. What did your team do that you feel is unique or something that the team is especially proud of (was there a big learning moment that the team had in terms of gaining knowledge of a new concept/process that was implemented).