OFET APPLICATION PROJECT PROPOSAL SUMMER 2023

Prepared for: OFET Database Project
Prepared by: Shubham V. More
Date: June 1, 2023

FOREWORD

Over part of the past Spring 2023 semester, I have been working with Rahul Venkatesh and Aaron Liu in order to devise a software development project that will aid research experimentalists. I have been given creative freedom on the technical side in terms of frameworks, build specs, and general design, to build an application as I see fit for the needs of this group. This document serves as a purpose to collect all my intuition about this project and provide a streamlined version to the thought process around this development project.

PROJECT PARTS

Part 1: Learning Process

In the learning phase of the OFET Data collection project and the new web application, it is essential to gain exposure to various technologies. Here is a breakdown of the big questions and technologies to explore:

1. Databases:
    - Relational vs. non-relational databases: Understand the differences between these two types of databases and their use cases.
    - PostgreSQL & MySQL: Learn about these popular relational database management systems and their features.
    - SQL language: Familiarize yourself with SQL, the language used to interact with relational databases.
    - Understanding current Python code and database interaction: Analyze how the existing Python code interacts with the database and retrieve relevant information.
2. ReactJS:
    - Basic concepts and components: Gain an understanding of the fundamental concepts in ReactJS and how components work together.
    - Virtual DOM: Learn about the concept of the Virtual DOM and how it optimizes rendering in React applications.
    - State management (Redux): Explore Redux, a popular state management library for managing complex application states.
    - Props: Understand how to pass data and properties between React components.
    - Lifecycle methods: Learn about the lifecycle methods available in React and how to utilize them for various purposes.
    - Event handling: Master event handling techniques in React applications.
    - Routing: Explore React Router for handling client-side routing in a single-page application.
    - Hooks: Understand React Hooks, which provide a way to use state and other React features in functional components.
3. Testing APIs with Postman:

    - Learn how to use Postman to write and test APIs. Postman allows you to send HTTP requests, analyze responses, and automate API testing.
4. Deploying the database to the cloud using Docker/Kubernetes in Azure:
    - Gain knowledge on containerization using Docker and orchestration with Kubernetes.
    - Explore Azure as a cloud platform and learn how to deploy the database using Docker and Kubernetes in Azure.
5. Implementing web scrapers/crawlers:
    - Learn about web scraping techniques and tools like BeautifulSoup or Scrapy to extract data from websites.
    - Understand how to write web crawlers that systematically browse and collect information from multiple web pages.
6. Implementing OpenAI public APIs for research aid:
    - Explore the OpenAI public APIs and their capabilities.
    - Learn how to integrate these APIs into the web application to leverage OpenAI's services for research assistance.

Part 2: Frontend Development:

1. Designing User Interface (UI):
    - Creating visually appealing and intuitive UI designs for the website.
    - Considering user experience (UX) principles to enhance usability.
    - Using design tools like Figma or Sketch to create UI mockups and wireframes.
2. Responsive Web Design:
    - Implementing responsive design techniques to ensure the website looks and functions well on different devices and screen sizes.
    - Utilizing CSS frameworks like Bootstrap or Material-UI for responsive layout and styling.
3. Interactive Components and Navigation:
    - Developing interactive and dynamic components using React or other modern frontend frameworks.
    - Implementing user-friendly navigation elements like menus, sidebars, or breadcrumbs.
4. Frontend Frameworks:
    - Utilizing frontend frameworks such as React or Angular for efficient and modular development of frontend components.
    - Leveraging libraries and tools from the chosen frontend framework to streamline development.

Part 3: Backend Development:

1. API Development:
    - Creating APIs (Application Programming Interfaces) to facilitate communication between the client-side and server-side of the web application.
    - Utilizing technologies like Node.js and Express.js for building robust and scalable APIs.
2. Database Setup and Management:
    - Setting up and configuring a PostgreSQL database to store and manage application data efficiently, potentially we could require multiple databases for easier access and control
    - Designing database schemas and ensuring data integrity through proper table relationships and constraints.
3. Web Crawlers:
    - Developing web crawlers using frameworks like Scrapy or libraries like BeautifulSoup to gather relevant information from external sources, if required for the project.
4. Authentication and Authorization:
    - Implementing authentication and authorization mechanisms to ensure secure access to the website and its features.
    - Utilizing tools like JSON Web Tokens (JWT) or OAuth for user authentication and session management.
5. Server-side Logic:
    - Writing server-side logic to handle user requests, process data, and perform necessary operations on the database.
    - Implementing functionality such as submission copy and other business logic specific to the project.

Part 4: Data Visualization

1. Data Plotting for Admin Accounts:
    - Creating a dedicated interface to plot specific data points against each other.
    - Implementing interactive charts and graphs to visualize the data in a meaningful and easily understandable way.
    - Providing customization options for admins to select and manipulate data variables for plotting.
2. Database State Viewing:
    - Developing an interface to view the current state of the PostgreSQL database.

Part 5: Integrating OpenAI API To be continued……

BASE OVERVIEW

The OFET Database project currently relies on users to manually input various pieces of data into an Excel spreadsheet connected to a PostgreSQL server for data input and storage. While this system has served its purpose to some extent, it presents several limitations and challenges that hinder the efficiency and effectiveness of the data collection process. However, the two most prominent issues that significantly impact the project are as follows:

Limitation #1: Fragmented Process

The current data entry process lacks cohesion and efficiency, as it involves fragmented steps that are not streamlined. Currently once the user inserts data points in the Excel sheet for each organic device they investigated the Georgia Tech team parses this information into CSV and JSON files, which were further parsed to create interconnected maps and tables for building the PostgreSQL database that stores the data. Unfortunately, this process is excessively lengthy, far from user-friendly, and poses challenges. While data parsing serves a purpose, relying on Excel sheets for creating CSV files, followed by additional parsing to generate tables, lacks scalability. It becomes increasingly difficult for research groups to conduct experiments without an easily accessible comprehensive list of past experiments or an efficient means of tracking them. As a result of this it makes the project cumbersome to manage and hinders the research progress as well.

Limitation #2: Data Validation & Management

The current data input process in the OFET Database project relies on manual entry into an Excel spreadsheet, introducing inefficiencies and the potential for errors. During the spring of 2023, undergraduates had to:

1) Analyze lab notebooks and literature to find specific data
2) Manually enter that data into the spreadsheet

which is a time-consuming and error-prone process. For instance, when a user submits a data spreadsheet, it is stored in a BOX folder or on their computer, waiting for an experimentalist to approve the data submission. While second-level data authentication by an experimentalist is necessary, the absence of automated data validation and standardization mechanisms in the current system poses significant challenges.

The lack of robust validation checks makes it difficult to ensure the accuracy, consistency, and integrity of the inputted data. Inconsistent formatting, varying data types, and missing or incorrect values can undermine the reliability and usability of the collected data. Furthermore, the lack of standardization in data management can lead to discrepancies and difficulties in data analysis and comparison, as experimentalists may have different interpretations or approaches to data entry.

PROJECT FEATURES

So for this project I am proposing to create a full stack web application that will be built in JavaScript and Python using ReactJS(an open-source JavaScript library used to build user interfaces on the web). This will be connected to a backend server that will eventually be uploaded to the cloud using PostgreSQL which is a relational database known to be secure, scalable, and flexible. Specific to PostgreSQL it allows us to create user-defined data types specific to chemistry, easier API integration, ability to handle large amounts of data(OFET template submissions), and last but not least ability to conduct transactions between different accounts(user/admin) to approve OFET template submissions for management purposes(Data Validation & Management).

In its most basic form, I am transfering the existing Excel spreadsheet by simply copying and pasting its contents into a website for each sheet. This website will then insert the information into tables that are related to each other, thus creating the PostgreSQL database. Additionally, the website will incorporate several key features as outlined below:

MAIN FEATURES

1. User Authentication and Registration

The initial screen of the web application will serve as a user authentication homepage, allowing both general users and administrators to login and register. Registration will require providing basic information that will be collected only once and every OFET data submission will contain this information in its metadata, including the user's first name, last name, and email address.

2. Multi-Layer Data Authentication

The authentication process aims to ensure secure access to the system and differentiate between different user roles. Each user account can submit a temporary OFET data file, which will be forwarded to admin accounts for review. Admin accounts have the authority to approve these data submissions by simply checking a button. Once approved, the data will be securely inputted into PostgreSQL. This solves the issue of fragmentation discussed earlier.

3. Ability to see a comprehensive OFET submission list

Each user account can access a personalized page that displays all their individual OFET data submissions, providing a convenient reference for their past contributions. For admin accounts, a separate page will be available, presenting a comprehensive view of all OFET submissions from all users. These submissions will be displayed in chronological order, with the newest submissions appearing first, facilitating easy tracking(fragmentation). Additionally, the admin account will feature an additional dashboard specifically designed to manage new OFET submissions that require approval. Administrators can efficiently navigate through the submissions, assess their content, and make informed decisions. This ensures timely approval and inclusion of high-quality data into the database.

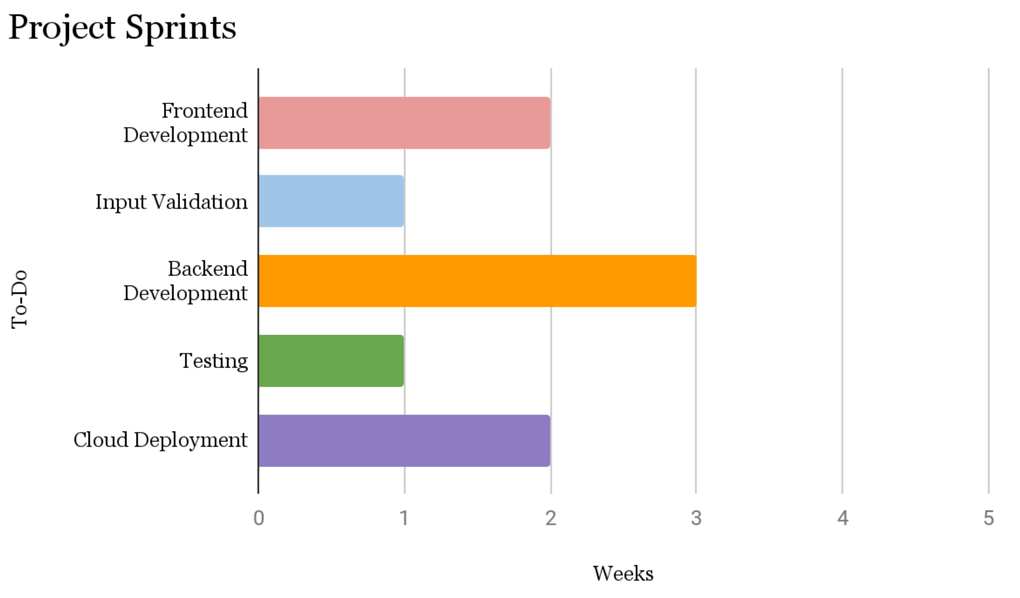
4. Iterative Experimentation Process

In the OFET Database project, experimentalists follow a repetitive process where they make incremental changes to evaluate the mobility of the device. While the steps remain largely the same, the key difference lies in modifying a single step at a time. To facilitate this workflow, a feature("create copy" button in the home page) will be implemented to enable users to recreate the procedure by making copies of the original submissions and modifying an individual step before pushing a new data submission for approval again. This functionality eliminates the need to re-type the entire procedure from the beginning. By duplicating the original submission, users can easily identify and modify the specific step they intend to change. The unintended positive consequence of this feature is that it empowers experimentalists to focus on testing specific factors while maintaining a standardized framework and not to mention extremely time efficient.

5. Data Visualization Feature

The admin account will have access to a dedicated page within the web application that automatically generates various types of graphs. This feature aims to assist administrators in visualizing the current state of the data. While the specific content of these graphs is yet to be determined, the DIAO & Georgia Tech group management will provide the necessary information. By leveraging visual representations, administrators can easily identify trends and patterns that may impact specific factors of interest.

PROJECT TIMELINE


Project Sprints
To-Do
Frontend Development
Input Validation
Backend Development
Testing
Cloud Deployment
Weeks