Product Design

Team Number: 30
Project Number: 37
Team Members:

- Tanmay Goyal
- Soveet Nayak
- Lakshmi Girija
- Imami Syed

Design Overview

Architectural design

Program structure:

Frontend : UI to detect anomalies; Should be able to login, register, upload CSV files, visualize data, edit data, view data and detect abnormalities according to users specifications.

- App.js : To run the frontend
- Components :
    - Navbar.js : To create a navigation bar for ease of access
    - Login.js : Functionality accessed from navbar and used to provide security for using the interface
    - Register.js : Functionality accessed from navbar and used to register a new user into the database
    - Dataupload.js : Accessed from the dashboard and is used for uploading the data into database
    - Abnormalitydet.js : Accessed from the dashboard and retrieves the uploaded data and according to users specifications, runs abnormality detection algorithms.
    - Graph.js : Accessed from the dashboard and is used to visualize data and abnormalities in the data
    - Dataedit.js : Accessed from the dashboard, it retrieves data according to the user's specs and gives an option to edit/delete the data.
    - Dataview.js : Accessed from the dashboard, used for viewing data uploaded previously.
    - Dashboard.js : All the functionalities can be accessed from the dashboard.

Backend : Connecting to the database to detect anomalies, login, logout, register, uploading CSV, editing the CSV, fetching the data, detecting anomalies.

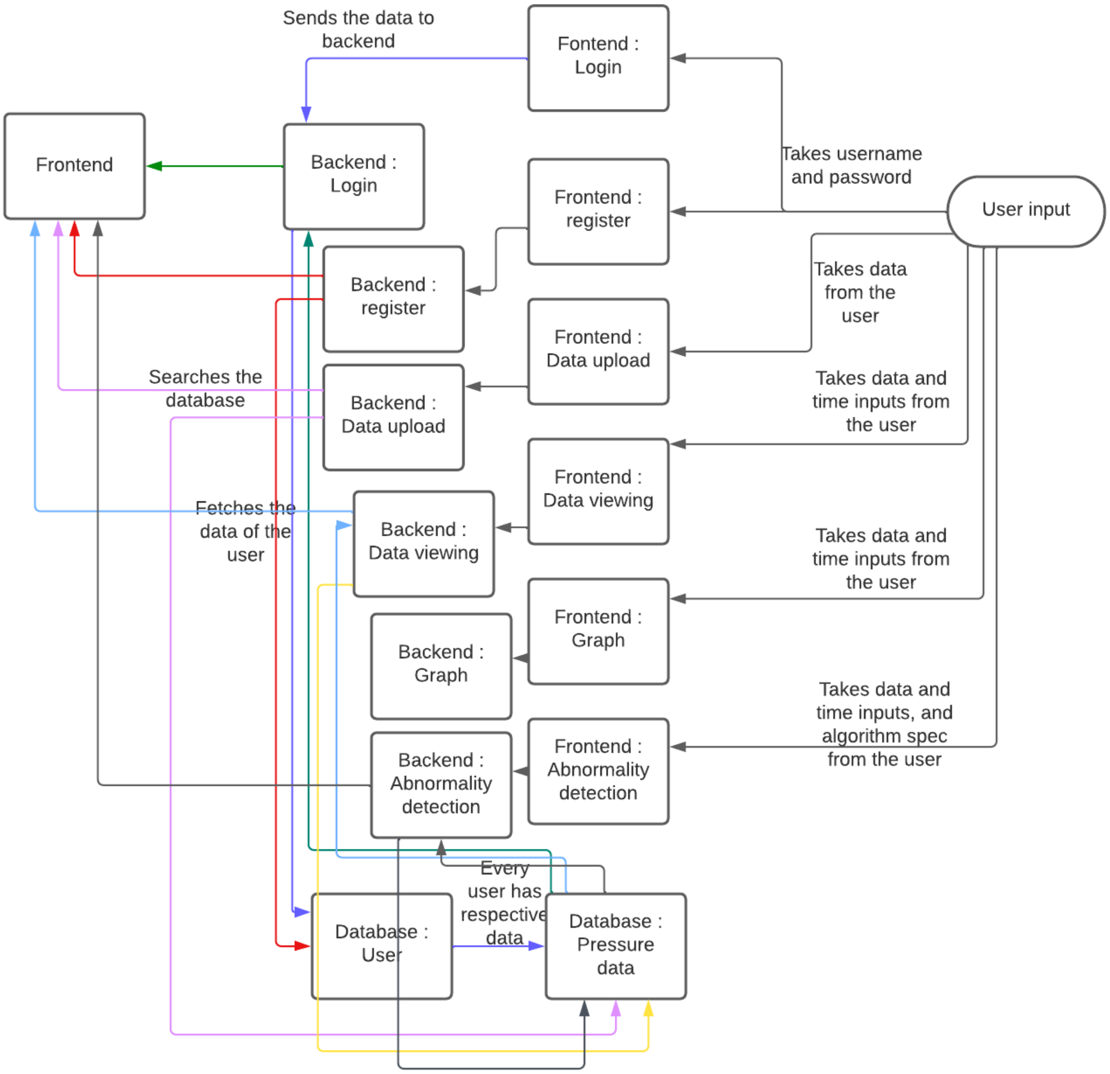
- App.py : To connect the frontend and the backend and perform all the above specified functionalities.

MongoDB : Specifically used for storing the data

- MongoDB is accessed from Mongo Compass locally.

Data Analysis :

- CSV file input
- CSV file analysis
- QuantileAD
- ThresholdAD


Sends the data to backend
Fontend : Login
Frontend
Backend : Login
Takes username and password
Frontend : register
User input
Backend : register
Takes data from the user
Frontend : Data upload
Searches the database
Backend : Data upload
Takes data and time inputs from the user
Frontend : Data viewing
Fetches the data of the user
Backend : Data viewing
Takes data and time inputs from the user
Frontend : Graph
Backend : Graph
Takes data and time inputs, and algorithm spec from the user
Backend : Abnormality detection
Frontend : Abnormality detection
Every user has respective data
Database : User
Database : Pressure data

Link to diagram:
https://lucid.app/lucidchart/0959bdc2-1932-4108-8dc0-971288e60be5/edit?invitationId=inv_195

a8ca0-7b54-4afc-b4f4-d644c1c6f501

System interfaces

User Interface

- The Users will be able to register themselves on the website using a register button. This register button will redirect them to a page where they will be asked to enter their details.
- The Users will be able to login into the dashboard using their own credentials. For this a login button will be placed which will redirect them to a page asking for their credentials.
- The Users will be able to enter huge amounts of data through CSV files. This data will be of the sensors in a water distribution system monitoring the flow and pressure. The UI for this purpose would be a “choose file” button followed by an “upload” button. The former would allow the users to choose any file they want. The “upload” button would send the data from the file to the backend which will ultimately get stored in a mongo database.
- A generate graph button will enable the users to visualize the data sent by them. In addition, a “show data” button will be present, which will show all the data uploaded by the logged in user.
- Additional buttons will be present which can be used by the users to set the threshold values. At least two buttons “set minimum” and “set maximum” will give the users an option to decide upper and lower thresholds. Any data beyond these limits will get registered as anomalies and will be displayed to the user on pressing the “show anomalies” button.
- The user can also choose from multiple algorithms like ThresholdAD and QuantileAD.
- The user can also view graphs from previous months.

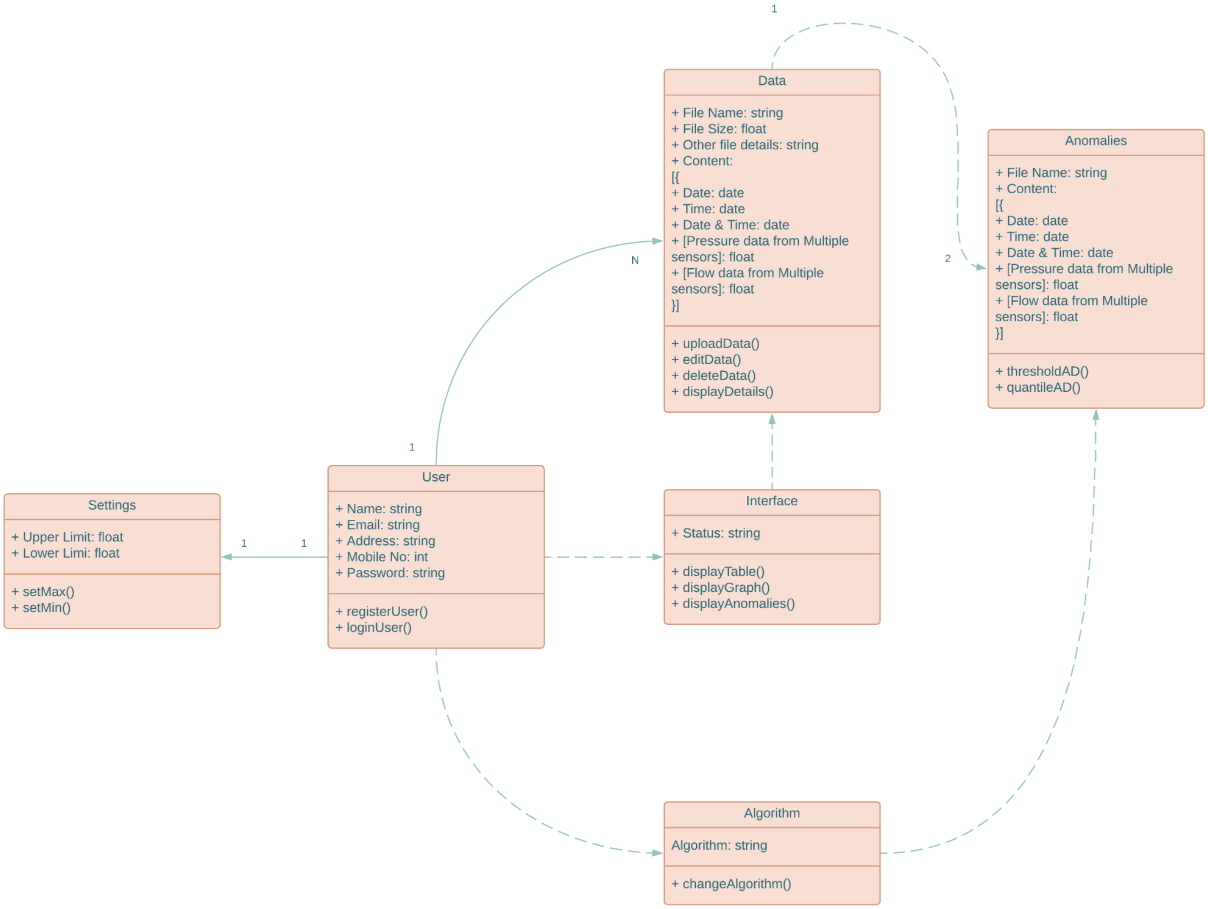
APIs

- We are using REST API’s for transferring information from frontend to backend and vice versa.
  - POST requests:
    - upload_data(): Data is accessed from the CSV file uploaded by the user. This data is then inserted into the mongo database one by one using insert_one function.
    - update_min(): Update the minimum limit for data to not have an anomaly. The settings of that particular user will get updated.
    - update_max(): Update the maximum limit for data to not have an anomaly. The settings of that particular user will get updated.

- GET requests:
    - get_data(): All the data pertaining to the logged in user is accessed using find function. This data is then returned to the frontend calling function in json object format.
    - get_min(): Get the lower threshold limit for detecting anomalies in data.
    - get_max(): Get the upper threshold limit for detecting anomalies in data.

Model

Class diagram :



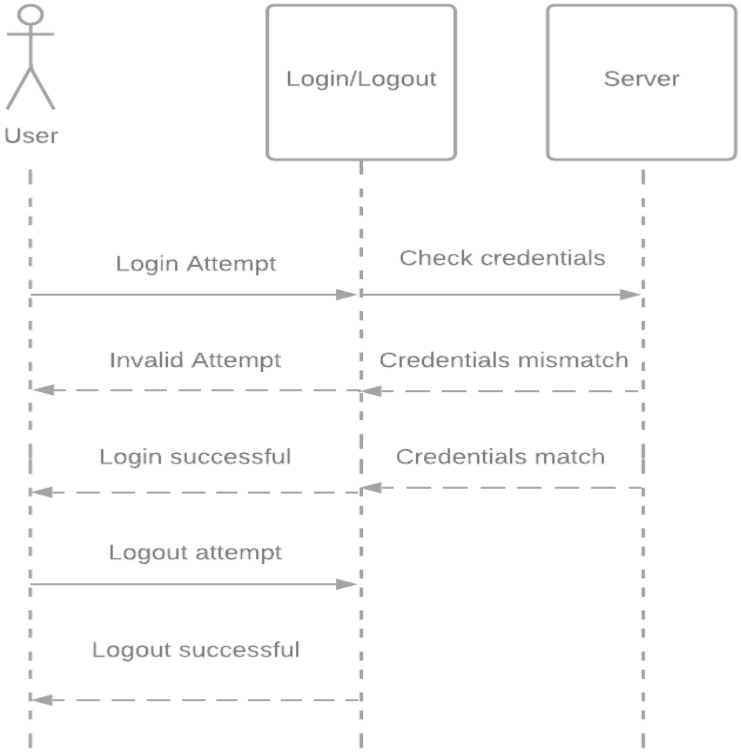
Link to diagram:
https://lucid.app/lucidchart/62e4041a-e0b4-47b1-b13f-ab2fdd5a8237/edit?invitationId=inv_77a9a6b6-3b38-42da-a7bd-01bc657a5385

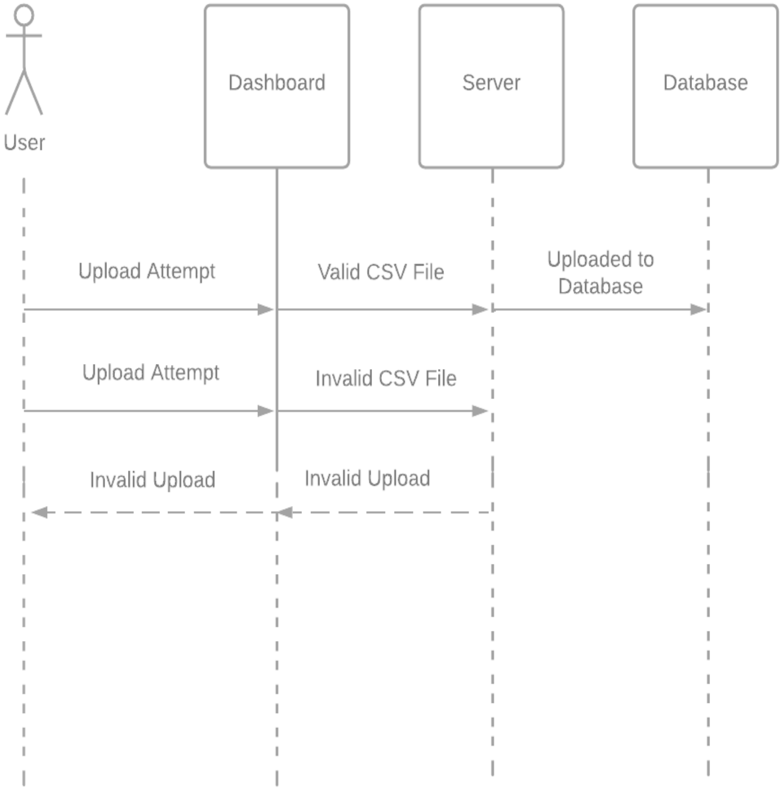
User	Class state: ● The user will contain all the data regarding the user such as name, email, address, mobiles, password. Class behavior: ● registerUser() : It registers the user and stores all its attributes ● loginUser(): takes the unique email address and password from the user and redirects the user to the User page
Settings	Class state: ● It is responsible for maintaining upper limit and lower limit of the algorithm implemented Class behavior: ● setMax() : sets the upper limit of the ranges in the thresholdAD algorithm ● setMin() : sets the lower limit of the ranges in the thresholdAD algorithms
Data	Class state: ● It stores the data from the sensors which include pressure values, date, time, flow data Class behavior: ● uploadData(): uploads the data simultaneously using the given inputs ● editData(): edits data depends on the changes made from time to time ● deleteData(): Deletes the particular data by taking time, date and sensor name and removes the data

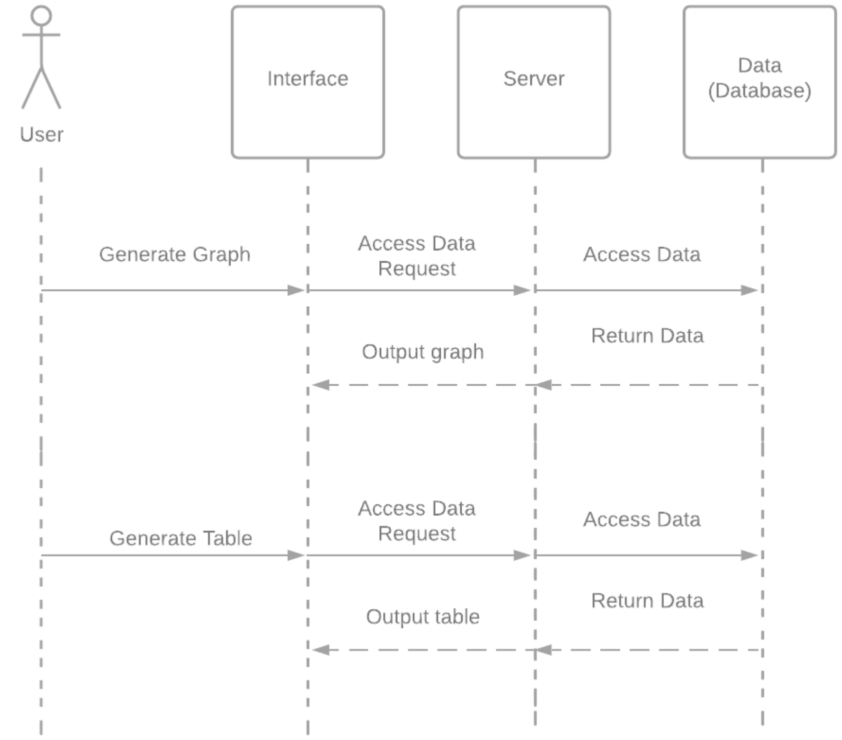
Interface	Class state: ● It is used for the display of Table , Graph, and anomalies on the data Class behavior: ● displayTable(): displays the table in the interface ● displayGraph(): display the graph in the interface ● displayAnomalies(): displays anomalies in the data anomalies
Algorithm	Class state: ● Stores the data of different algorithms Class behavior: ● changeAlgorithm(): allows us to change between multiple algorithms
Anomalies	Class state: ● Used to display anomalies in the data. Class behavior: ● thresholdAD(): Algorithm to detect anomalies by setting the upper and lower thresholds ● quantileAD(): Algorithm to detect anomalies using quantiles

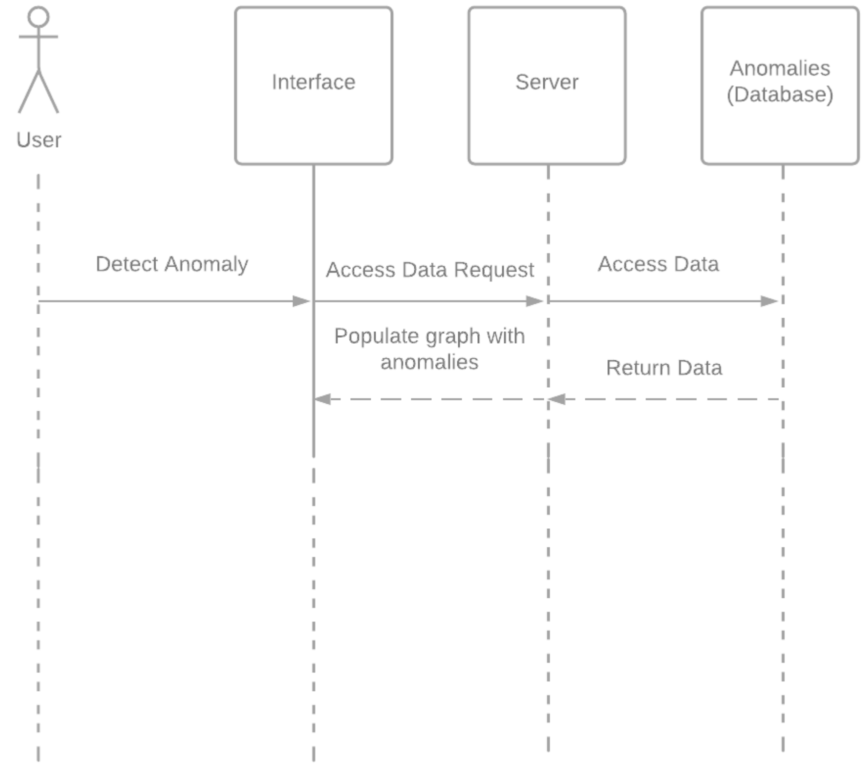
.

Sequence Diagram(s)


User
Login/Logout
Server
Login Attempt
Check credentials
Invalid Attempt
Credentials mismatch
Login successful
Credentials match
Logout attempt
Logout successful


User
Dashboard
Server
Database
Upload Attempt
Valid CSV File
Uploaded to Database
Upload Attempt
Invalid CSV File
Invalid Upload
Invalid Upload


User
Interface
Server
Data (Database)
Generate Graph
Access Data Request
Access Data
Output graph
Return Data
Generate Table
Access Data Request
Access Data
Output table
Return Data


User
Interface
Server
Anomalies (Database)
Detect Anomaly
Access Data Request
Access Data
Populate graph with anomalies
Return Data

Design Rationale

- 22-02-2022:
    - How to store data in MongoDB: Storing the polished CSV data using TimeSeries or not. Currently data is being stored as documents in the collection which is not the best method. If the data has a frequency (have to look into it and be advised by the client), we can store it in Timeseries collection in MongoDB.

    - Graphs were previously implemented using the devexpress module. They need to be implemented using the Plotly.js module for better experience and more interface options for the user.

    - Add a Generate graphs button for generating the graph on demand which was previously shown directly on uploading the CSV file.