Allergy Prediction Using Artificial Intelligence

GROUP SDMAY24-13

Eric Christensen, Zoe Davis, Josh Dutchik, Blake Friemel,
Jack Gray, Michael Koopmann, Jihun Yoon

IOWA STATE UNIVERSITY

Introduction – Who are we?

- Eric Christensen – Database Manager
- Zoe Davis – Documentation and Team Organizer
- Josh Dutchik – Frontend Designer and Backend Support
- Blake Friemel – Documentation and Client Interaction
- Jack Gray – Cloud and Systems Engineer
- Michael Koopmann – AI Development
- Jihun Yoon – Documentation and Backend Support

Introduction – What is our project?

- The goal of this project is to develop an advanced healthcare application using AI to predict allergens that can be used by a doctor to improve a patient's treatment plan
- By analyzing a patient's medical information, known allergies, and symptoms, the system predicts allergens and products that patients will likely be allergic to
- This system will be help doctors prescribe safe products and prevent adverse reactions

Problem Statement

Problem

- Medical care is hard to optimize for every individual (unique patient variables and medical history)
- Allergic reactions can be unpredictable and pose unforeseen risks in treatment

Solution

AI allergen prediction application

- Efficiency in diagnosis
- Increased accuracy/reduced errors
- Wider availability
- Remote diagnosis and monitoring
- Non-invasive testing

Project Components

- Frontend – Website (React application)
  - Patient and doctor users
- Backend – Node.js Server
  - Managed through MySQL and node.js
- Database – Amazon RDS Database
  - Stores three tables: Doctor, patient, and products
- AI Model – Built using Keras and TensorFlow libraries.
  - Stored on an S3 Bucket
  - Trained using an obfuscated dataset
  - Runs patient data to return likely allergens

Functional Requirements

- Website must allow patient users to...
  - **Navigate** to the survey
  - **Input** their typed data into the survey
  - **Select** from the options provided by the UI
  - **Submit** the survey
- Website must allow doctor users to...
  - **Input** typed data (username, password, patient name, etc)
  - **Login** to their account
  - **Search** for a patient
  - **Run** analysis of allergens

Functional Requirements

- The website itself must...
  - **Display** an interactable GUI
  - **Output** patient data
  - **Output** predicted allergens
  - **Communicate** with the backend to display relevant and accurate pages/information
  - **Be hosted** on an EC2 instance and Google VM instance

Functional Requirements

- Backend (Node.js Server)
  - Send and receive HTTP requests to and from the Amazon RDS database
  - Send and receive JSON file format to and from the model
  - Function calls triggered by frontend
- Database (Amazon RDS Database)
  - Store patient, doctor, and product tables
  - Send and receive HTTP requests to and from the backend

Functional Requirements

- AI Model
    - Trained using an excel file
    - Input and output JSON files
    - Use rules of association to predict potential allergic reaction
    - Output ingredients with over 70% likelihood of allergic reaction
    - Output products that contain high-risk ingredients
    - Must be stored on an S3 bucket to be pulled by backend

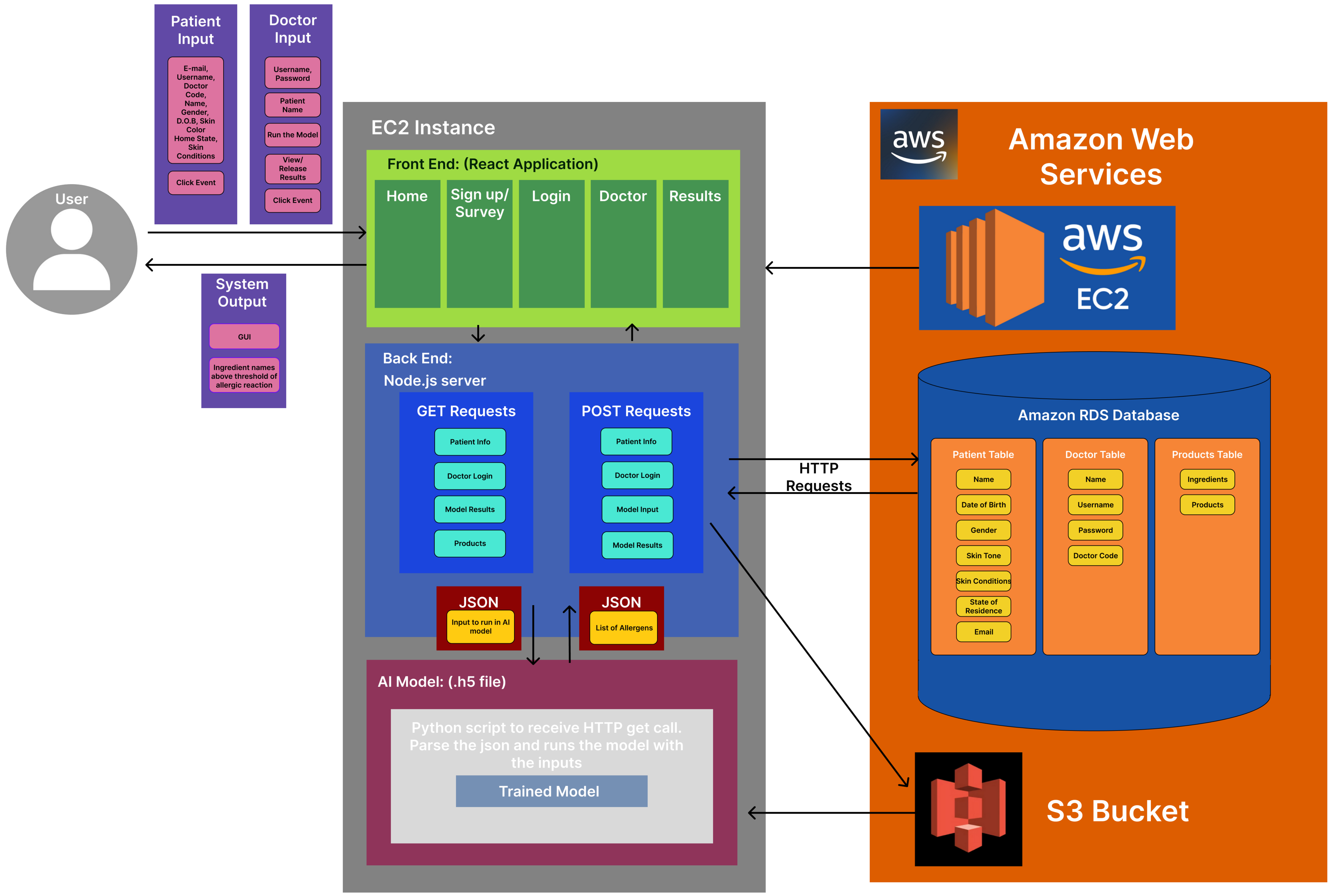
Non-Functional Requirements

- Website
    - Intuitive and easy to navigate
    - Survey reduces the amount of variability added to the data by formatting certain inputs
    - Doctor accounts accessible to only medically licensed individuals
    - Reliable and have little downtime
    - Accessible from anywhere in the United States
    - Accurate and relevant information
    - Aesthetically pleasing (we are not design students)

Non-Functional Requirements

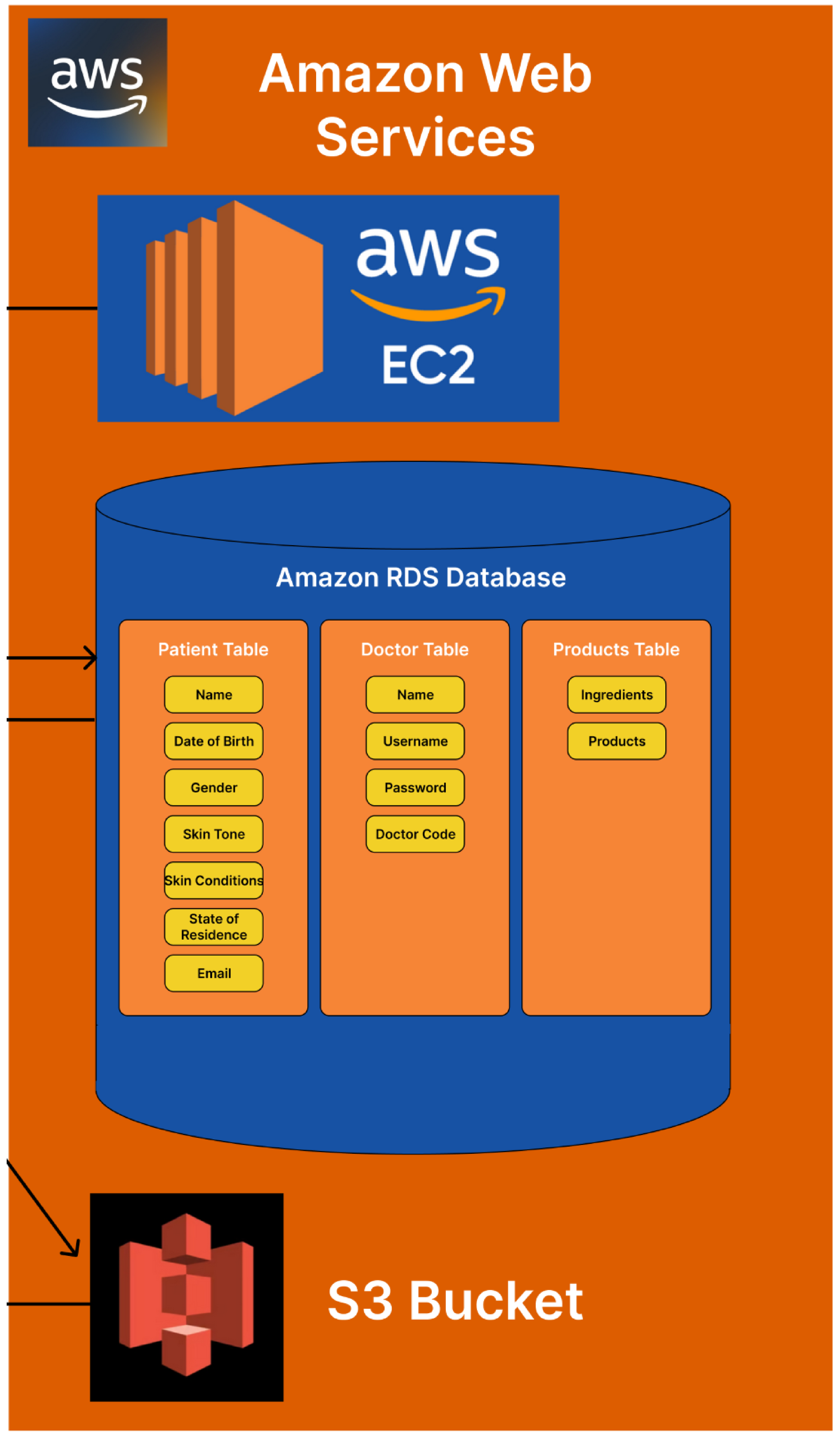
- Backend
  - Communicate promptly within a short period of time
- Database
  - Fields and tables should be clear and related to their stored variables
  - Secure and require proper authentication and authorization
  - Scalable in both vertical and horizontal dimensions
  - Reasonable response time
- AI Model
  - Maintains high level of prediction accuracy
  - Retrainable
  - Returns results in a timely manner (10 seconds or less)

Overall Project Visual Design



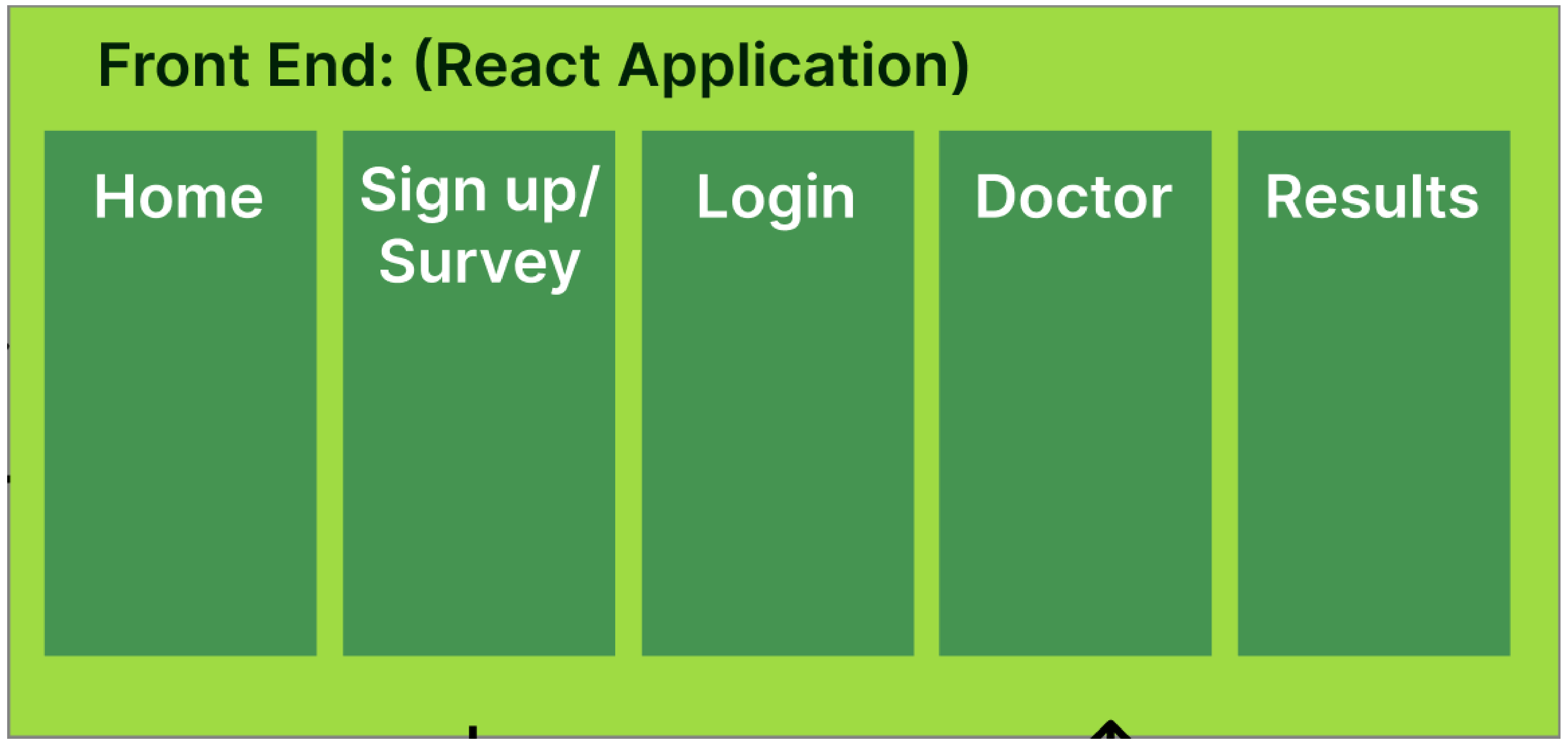
Amazon Web Services

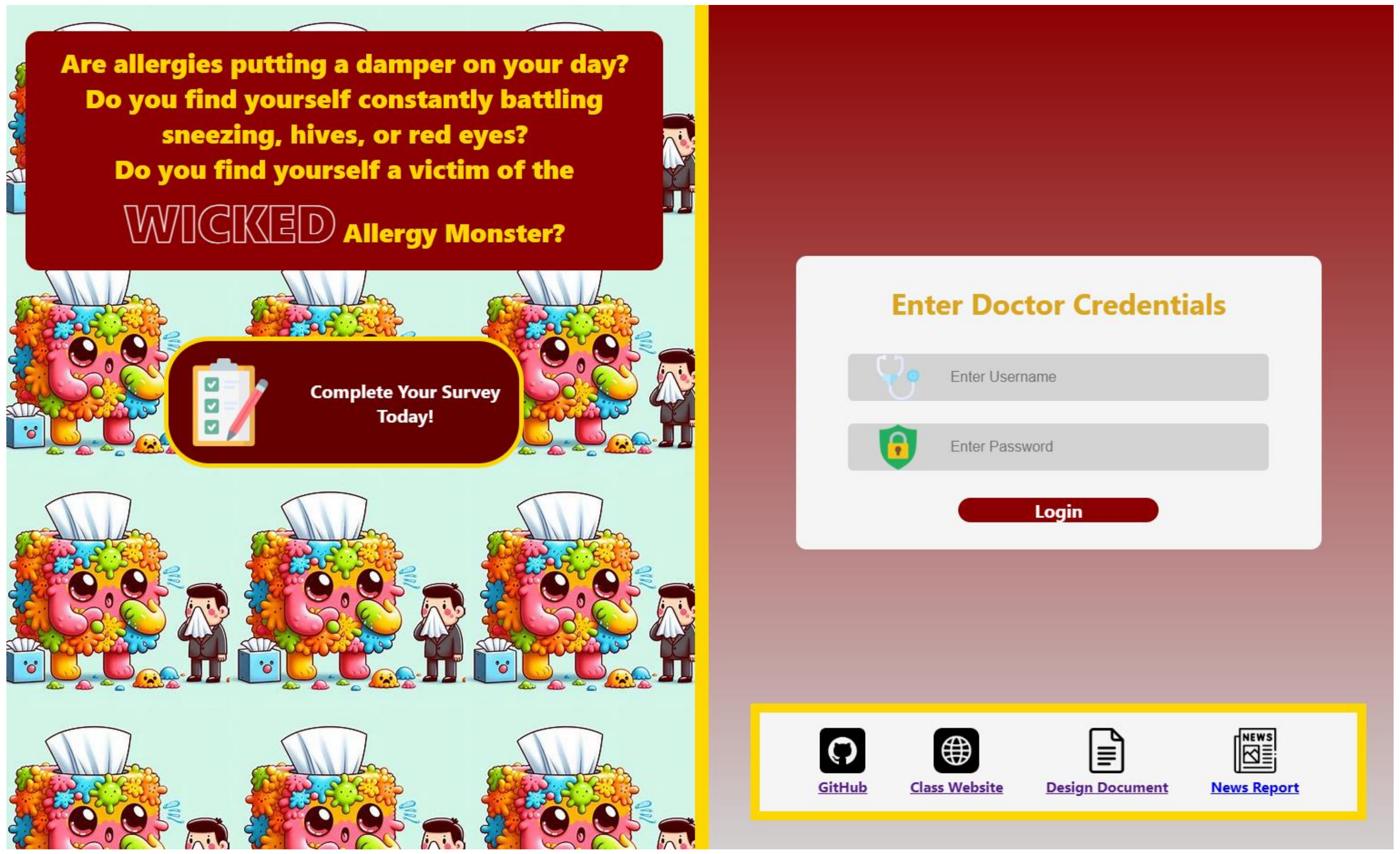
- Frontend, backend, and AI model is hosted entirely on EC2 instance
  - (GCP instance for comparison)
- Utilized Amazon RDS for our database
  - Stores patient, doctor, and products table
- S3 Bucket to store model
  - Size issues



Frontend Design

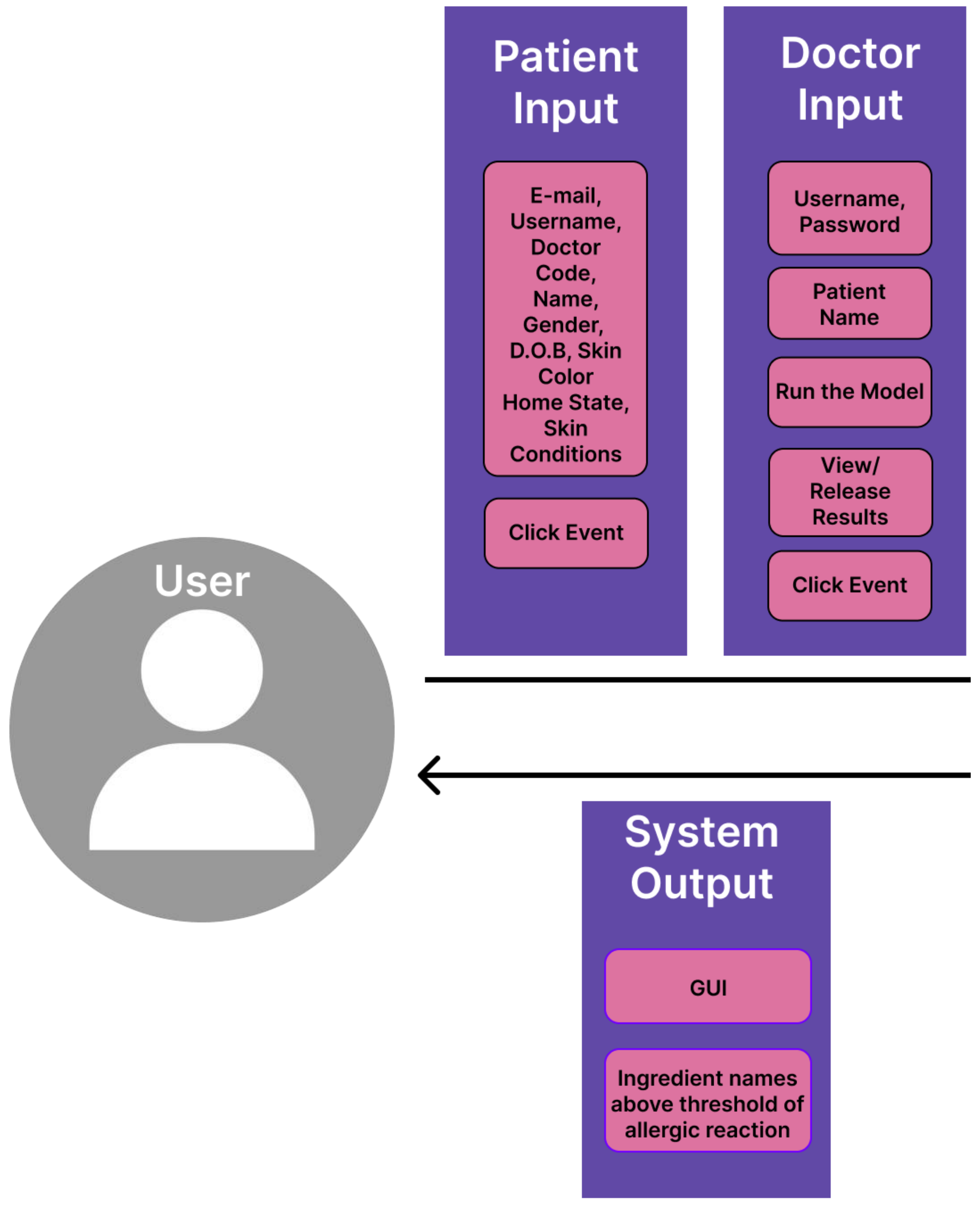
- React Application
- Changes variables and triggers functions in backend
- 5 Elements
  - Home
    - Navigate to log in and survey
  - Sign up/Survey
    - Patient inputs data to be run in model
  - Login
    - Checks doctor username and password -> navigates to doctor
  - Doctor
    - Doctor can search for patients and view patient data. Navigates to results
  - Results
    - Can cross references allergens with commonly used products for a list of products/medications to avoid

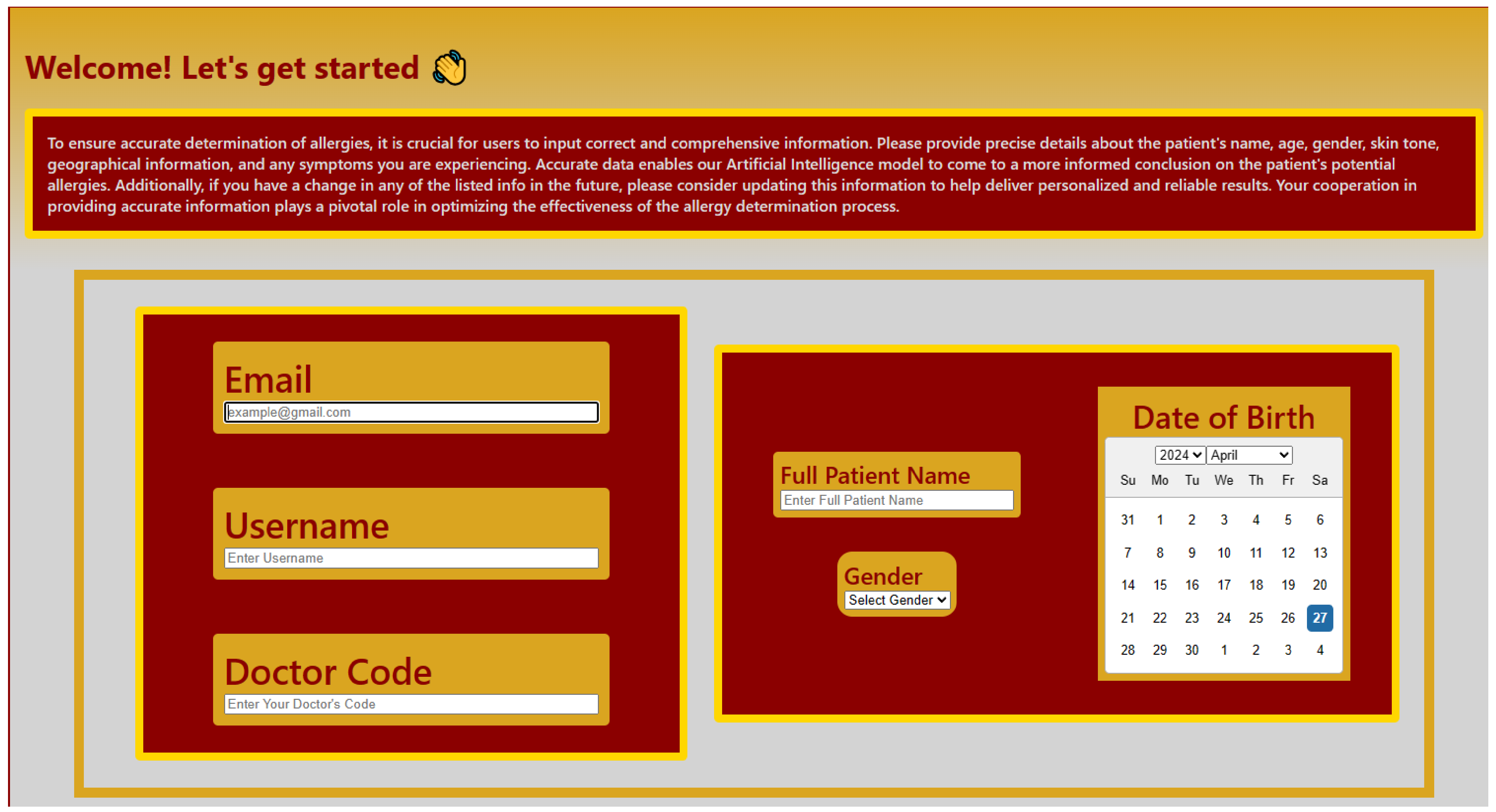



Are allergies putting a damper on your day?
Do you find yourself constantly battling sneezing, hives, or red eyes?
Do you find yourself a victim of the WICKED Allergy Monster?
Complete Your Survey Today!
Enter Doctor Credentials
Enter Username
Enter Password
Login
GitHub
Class Website
Design Document
News Report
Home page

User Interaction

- Patient User
  - Completes survey
    - Inputs: E-mail, username, doctor code, name, gender, D.O.B., skin tone, state of residence, skin conditions
- Doctor User
  - Logs in
  - Searches for patients
  - Views patient data
  - Runs product analysis
  - Signs out


Patient Input
E-mail, Username, Doctor Code, Name, Gender, D.O.B, Skin Color Home State, Skin Conditions
Click Event
Doctor Input
Username, Password
Patient Name
Run the Model
View/ Release Results
Click Event
User
System Output
GUI
Ingredient names above threshold of allergic reaction


Welcome! Let's get started 👋

To ensure accurate determination of allergies, it is crucial for users to input correct and comprehensive information. Please provide precise details about the patient's name, age, gender, skin tone, geographical information, and any symptoms you are experiencing. Accurate data enables our Artificial Intelligence model to come to a more informed conclusion on the patient's potential allergies. Additionally, if you have a change in any of the listed info in the future, please consider updating this information to help deliver personalized and reliable results. Your cooperation in providing accurate information plays a pivotal role in optimizing the effectiveness of the allergy determination process.

Email: example@gmail.com

Username: Enter Username

Doctor Code: Enter Your Doctor's Code

Full Patient Name: Enter Full Patient Name

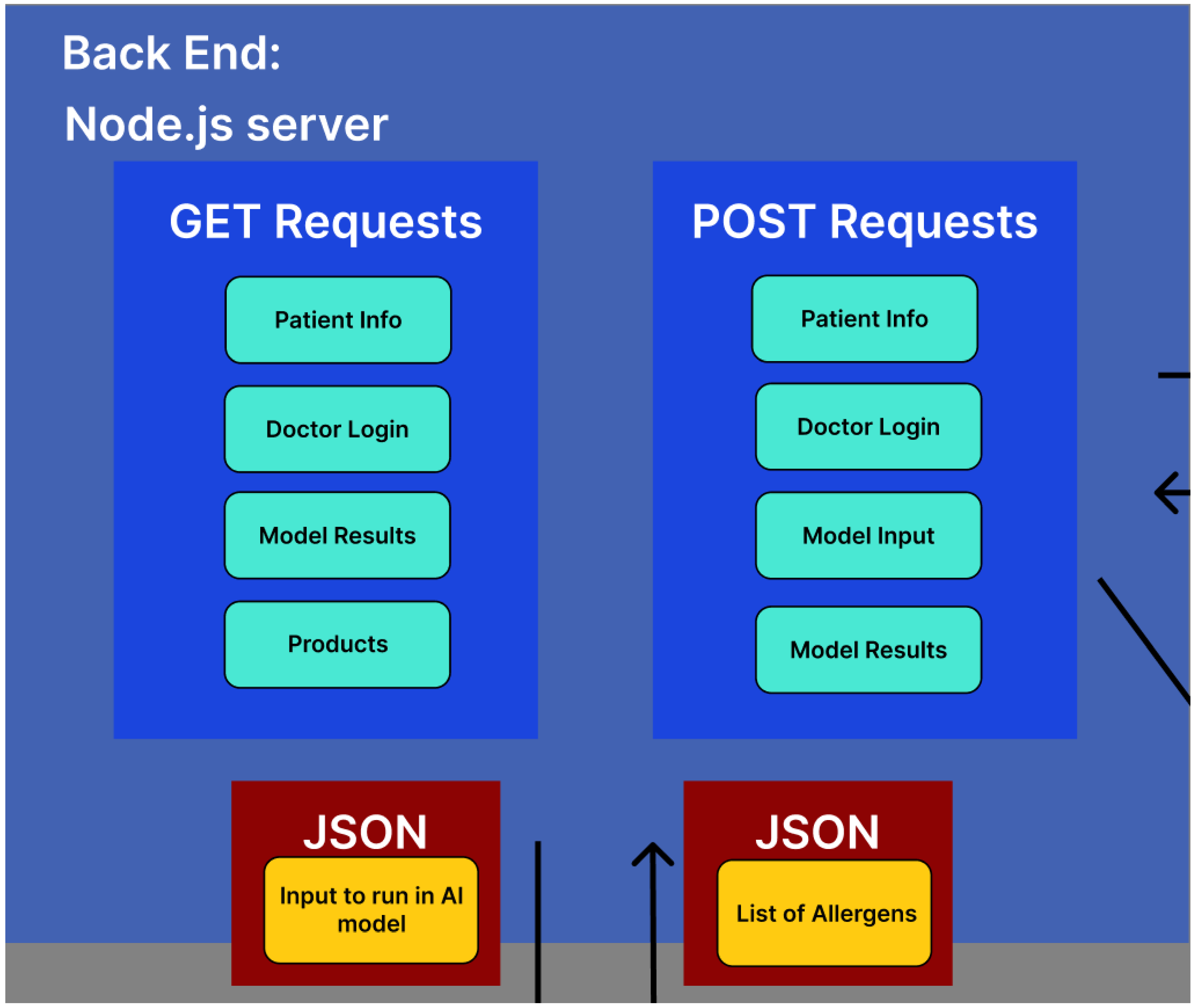
Gender: Select Gender

Date of Birth: 2024 April

Survey page

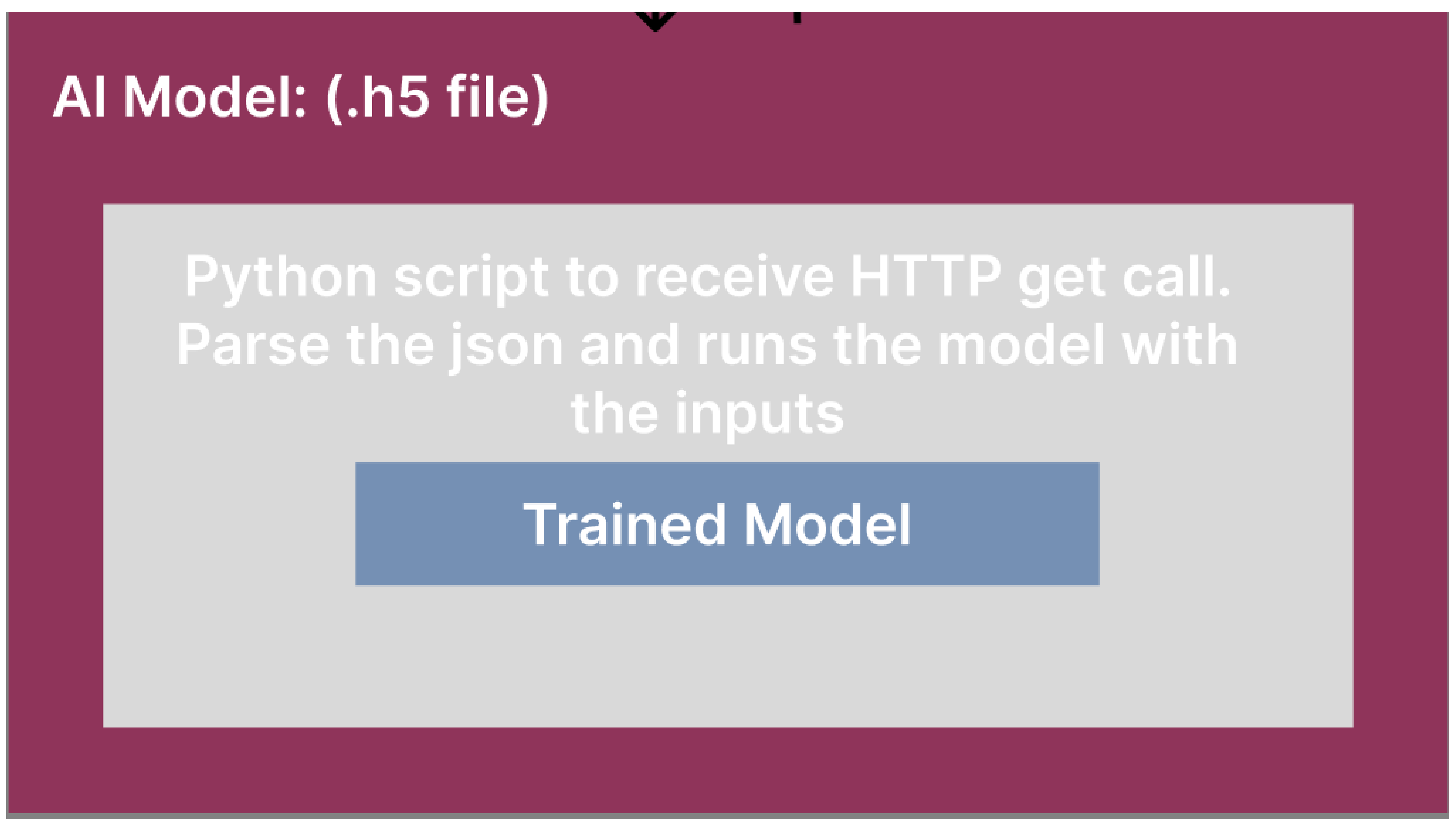
Backend Design

- Node.js Server
- GET, POST, DELETE requests
  - HTTP requests to database
- Pulls AI Model using S3 Bucket
  - JSON file format to and from model



AI Model Design

- Built, compiled, and saved using Keras and TensorFlow libraries
- Jupyter Notebook
- Inputs patient data
- Outputs percentages of allergens
- Stored on an S3 Bucket
- 70% threshold (Changeable)


AI Model: (.h5 file)
Python script to receive HTTP get call. Parse the json and runs the model with the inputs
Trained Model

IOWA STATE UNIVERSITY

Project Demo

Home
Gaming
Record & Stream
Performance
Smart Technology
Search
Now Recording
PRESS CTRL + SHIFT + E TO
Record
Live Stream
Scene Editor
Media
Settings
Controls
Fullscreen
Region
Active Window
Start Recording
Microphone
Ctrl+Shift+M
Camera
Ctrl+Shift+C
Microphone Level
50%
Push to Talk
Mouse Middle
Show Indicator
Disabled
Record Desktop
Enabled
Preview
RECORDING IN PROGRESS
RECORDING
Now Recording
00:00:00
00:00 / 00:09
unknown_2024.04.27-22.29.mp4
00:00:00
Activate Windows
Go to Settings to activate Windows
Type here to search
61°F Cloudy
ENG US
10:30 PM
4/27/2024

Frontend Testing

- Postman
  - Verify the functionality from the frontend to the backend
  - Verify the functionality and results of our API
  - Directly interact with our database through HTTP requests
- React Developer Tools Extension
  - Useful to see status of requests directly on web browser
  - Allows us to efficiently inspect component heirarchy

Backend Testing

- Console logs would indicate where problems were occurring and if anything went wrong
- For each request we designed use cases for each possible outcome
- Postman Requests
  - Similar to the frontend, we designed test suites for each request that was being made

Challenges and Solutions

- Connecting the EC2 Instance with the Backend
  - Could not make requests to the EC2 instance to POST data to our database
  - Solution: Corrected The URLs and Ports, corrected the package.js
- EC2 Instance ran out of storage
  - Installing tensor flow would make the instance crash
  - Solution: Updated our EBS volume to have more storage

Questions?