5 Testing

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, power system, or software.

The testing plan should connect the requirements and the design to the adopting test strategy and instruments. In this overarching introduction, given an overview of the testing strategy. Emphasize any unique challenges to testing for your system/design.

5.1 Unit Testing

- Data Integration Units
    - Units to test: Data import, data processing, data security
    - Testing Approach: Providing sample data and verifying that the system correctly imports, processes, and secures data
    - Tools: Python (PyTest) or JUnit, Postman
- Allergy Reaction Prediction Units
    - Units to test: Reaction prediction accuracy
    - Testing Approach: Use cases with known allergens and medical histories to verify accuracy of reaction prediction
    - Tools: Testing scripts/frameworks
- User Interface Units
    - Units to test: User interface functionality, UX, UI design consistency
    - Testing Approach: Test UI for functionality with simulated user interactions; Test UX through user feedback; Test UI design for consistency with design guidelines
- Ethical and Legal Units
    - Units to test: Data privacy, security, patient confidentiality
    - Testing Approach: Test if the system complies with privacy and security regulations relevant to project and if the system maintains patient confidentiality
    - Tools: Manual reviews, security scanning tools
- Performance and Scalability Units
    - Units to test: System performance and scalability
    - Testing Approach: Test system performance under load, test scalability as data and user loads increase
    - Tools: Custom scripts, JMeter
- Error Handling Units

- Units to test: Error detection and reporting
- Testing Approach: Introduce errors and exceptions to see if system can identify and handle them appropriately
- Tools: Custom error scripts, logging tools for error detection

5.2 Interface Testing

Since our project focuses on the training and learning of our AI (Artificial Intelligence) model our interfaces are limited to the user experience and interaction. Most of the heavy work is done behind the scenes but it does not mean our interfaces should be ignored and not focused heavily upon to ensure a quality and safe experience. Our users will need to be able to plug in, access, and retrieve their data. This experience is our interface. Users will need to be sure that our project handles their medical history and data safely and that the experience is quick and easy. Since a major goal of this project is to change the way and total experience allergies are discovered it is important that we do just that by making the experience pleasant and safe.

Our main interface is our website which needs to be secure which users need to be able to use quickly and easily. The main composition of one interface is plugging in user information and medical history to have our AI model predict any allergens. This user experience and interface will need to be tested in three ways

1. The data that is being entered is correct and used properly

2. That data is secure and safely used avoiding a breach in security

3. Inputs are easy to understand and enter avoid little confusion

A second composition of our interface will be the actual results that are given to the user. Although not an overly complex interface it is still of the most importance that two criteria is met

1. The results are easily accessible and easy to read

2. The results are correct, and the AI model is producing the correct output, and the correct output is being sent to the user

3. The results are secure and safely stored in the websites data

Tools:

Postman- helps create, build, and use APIs, will be used for more generic testing making sure requests and data are being properly sent and received

Selenium- UI automation testing, will allow for testing of our interface across multiple browsers

WebdriverIO- small and lightweight component tests

Katalon Studio- uses other testing software and APIs but is a good place to have an “all-in-one" experience

5.3 Integration Testing

We will be using a hybrid integration testing, but it will mostly be bottom up.

At the current moment, there are only two critical paths. The first path is Amplify to Cognito for login authentication and Amplify to our AI model endpoint. These handle the requirements for

security and allowing user interaction with our AI model. New paths could be created depending on how we implement our backend with the data servers.

- Frontend/Web application to AI model instance. This path will go through multiple components being:
    - Amplify to LambdaFunction
    - LambdaFunction to SageMakerNotebook Endpoint (Kernel Gateway)
- We will first start the bottom up going with just LambdaFunction and our Notebooks Endpoint. This can be tested using just AWS SDK (software development kit). For this we'd create a simple python file to run locally. We'd create a client and some test data that we'd know the output from by running against the AI model locally. Then all we'd need to is connect our client to the lambda endpoint feed in the test data invoke the call and assert that the value returned is as we'd expect.
- Next, we will integrate Amplify into our testing. Our testing will be very similar to earlier testing of using test data for a call and then asserting that the response is the correct response both unit and value wise the only difference will be we will be using Amplify mock function to simulate our POST calls.
- Frontend/Web application to Cognito.
    - Amplify to Cognito
- For this testing we will be reusing the amplify mock functions to mock calls to Incognito. We will be testing the GET call of retrieving a user's identity token when they signin.

We do not need any real integration testing for our model training as this can all be done locally till you have your model, and then you need to host it on an instance with a gateway.

5.4 System Testing

The primary goal of system testing would be to validate that the system functions accordingly to our requirements, specifically with regards to the turnaround time and accuracy of results. Our system level testing strategy starts with module testing and ensuring each individual component of our design works. After having insurance that our system components work, validating the functionality and developing unit tests to test performance will be implemented. This will allow us to test edge cases and ensure that the information coming back from our system is in a timely manner. This will be tied to closely to the requirements for a majority of the tests but edge cases might deviate slightly from our requirements.

In regards to end-to-end testing tools, we plan on using JUnit to perform a lot of these tests since they can make testing fully automated. The specific unit, interface, and integration tests will be closely tied to performance. We plan on implementing unit tests to cover any variability in the AI, as well as making sure that it has an output as well. We also plan on using SQLUnit for the database which will help us isolate any potential problems we may face with undesirable results from our design.

To coordinate our test cases, we will use either TestRail or qTest to manage the organization of our test cases as well as generate test reports. The feedback from the tests will be crucial to the success

of our project in regards to maintaining timeliness as well as verification that the system and each individual component are responsive in an appropriate manner.

5.5 Regression Testing

To ensure that new additions do not break the old functionality, we employ version control and continuous integration practices through tools like GitHub.

- All new code will be pushed into its own branch, and when it's complete, the branch will request to merge into main
- Each merge request will require two reviewers to be able to be merged into main

We need to ensure the stability and performance of implemented critical features, particularly the diagnostic AI system, to prevent any degradation in its accuracy and functionality.

5.6 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional are being met? How would you involve your client in the acceptance testing?

Our team will create and maintain comprehensive documentation that clearly outlines the design requirements and how they are addressed in the system. This documentation will be used as a reference point for our client to ensure that the original requirements are met. In addition, we will create a prototype of our AI system, which will allow the client to interact with the system and verify that all functional and non-functional requirements have been met. This level of involvement from the client ensures that the final product aligns with their expectations.

5.7 Security Testing (if applicable)

Our project will require security testing since the system will be handling sensitive data like patient records which are required to be confidential. To test the security of our system, we will complete the following:

- Vulnerability Assessment
  - Scan the system for vulnerabilities and security issues
- Authentication and Authorization Testing
  - Ensure authentication and authorization of users is working correctly
- Data Encryption and Transmission Testing
  - Ensure that when sensitive data is encrypted at rest and transit that it's protected
- Security Compliance Testing
  - Assess if the system adheres to HIPAA and other relevant standard/regulations
- Session Management Testing
  - Verify sessions are secure and not easily exploitable
- Database Security Testing
  - Assess the security of the database to prevent breaches
- Security Awareness and Training
  - Train on security practices and relevant possible threats on the system
- Incident Response Testing
  - Prepare to respond to security incidents effectively
- Secure Configuration Testing
  - Ensure servers, databases, etc. are securely configured and are protected from common security threats.

5.8 Results

Unit testing will result in all unit's listed working properly and accurately so the AI system will run smoothly and provide accurate predictions to the user. Our unit testing plan ensures compliance by making sure the AI model and all other parts of the system will perform efficiently for the user while the AI accurately makes predictions.

Interface testing will result in the website interface being able to collect data and produce prediction results correctly, securely, and easily. Our interface testing plan ensures compliance for a secure, quick, and user-friendly experience for the user while also meeting the project's main goals of predicting allergy reaction to products.

Integration testing will result in all critical paths and components communicating properly with each other. This testing ensures compliance with requirements by verifying that the integration points of the AI model are working as expected and are meeting the security and functionality requirements.

System testing will result in better performance, accuracy, and robustness of the system overall, as well as specifically with the AI model. System testing is in compliance with the requirements of the project because it validates that the entire system is running as accurately and efficiently as possible, resulting in the allergy predictions made the AI to be more accurate.

GitHub will prevent regression by using a continuous integration that prevents new code breaking existing functionality. This is in compliance with our requirements because it ensures that new features and changes won't negatively impact the stability and performance of the allergy prediction.

Acceptance Testing will result in proving that our design for the allergy predicting AI is in compliance with our project requirements. Testing will go over the outlines of the design to make sure all requirements are included and will go over how the requirements are implemented into the system.

Security testing will result in all aspects of the security of the user and patient data used will remain secure and protected. This testing will be in compliance with our requirements by demonstrating the system's commitment to handling sensitive data securely and adhering to relevant standards and regulations, like HIPPA.

In conclusion, our comprehensive testing design validates the design of our allergy predicting AI aligns perfectly with the intended goals and requirements for the project. Our testing efforts will culminate in a robust and reliable AI allergy prediction system. The system will not only perform efficiently and accurately, but it also will uphold standards of security, usability, and compliance with our project requirements. As we move forward, the foundation of our design ensures a trustworthy experience for users.