3. Project Plan

3.1. Task Decomposition

- Predict whether a patient would have an allergic reaction to a medicine
    - Determine the tools and libraries to build out the AI prediction software
        - Deep Learning Libraries: Keras
    - Find an existing model since building one would take too long
    - Train the selected model with the data we find
    - Test if the accuracy (delta) is at the desired level, if not repeat the previous and current step
- Support processing of large input sets
    - Setup system environment in AWS
- Display prediction and confidence level clearly
    - Graphs and charts should be used to display basic numbers that are important to visually understand
        - Find a third-party graphing library / API to use that is compatible with our front-end language/framework
    - The media should be properly labeled and explained if necessary for transparency
- Location for user to upload information to test the model
    - User can easily fill out a form where they input the necessary information
    - Determine input file type (PNG, JPEG, etc)
- Create an application as an interface between our users and the AI model
    - front-end: React
        - Form which restricts user input to valid data
        - Deploy front-end to AWS
    - Backend: Lambda cloud functions written in Python for AWS
        - Create validation function for data
        - Function to send data between front and backend
        - Function to read/write from database
- The remaining UI should be visibly appealing to the user
    - Prototype a Figma design and mock-up
    - Define a color scheme to follow
    - Optimize loading time
- Collect data to train and test our model
    - Ensure data collection and storage does not violate any health privacy laws
    - Research which laws are relevant to our project
    - Collect data from advisor
- The model should be tested for an overall accuracy percentage to report
    - Define a percentage threshold to determine if our model is suitable for use
        - Use medical standards for effective diagnosis as a benchmark.
        - The threshold should be generous to start, then constrain it as we develop the project further
- Logs should be implemented to catch faults or errors
    - Logging tools implemented at front and back end
    - Errors give descriptive reasons as to what caused the error and remediation

    - Logs stored to a database or dedicated logging server such as Splunk
- File structure should be clear
    - Any files needed for the project are grouped together by how closely related they are
    - Data and code are in separate folders
- Code should be well documented
    - All necessary parts have descriptive comments
    - Detailed README file giving an overall view of how the project goes together
    - Possible UML diagrams to illustrate the data flow and end to end connections. Allows for easy high level understanding of our project for new users. (Sequence diagrams, class diagrams, etc)

3.2. Project Management/Tracking Procedures

Waterfall+agile project management best fits our project needs. Since many of the requirements can be developed independently of each other at first, it makes the most sense to work on them concurrently. It also keeps everyone working on something as new stories can be pulled in as needed. Using the waterfall style would force all of us onto one component at a time, and the work may not divide evenly between everyone. Waterfall alone also is missing the critical reflection step that agile has, meaning there is no real discussion for what is going well and what is not. Likewise, a fully agile management style uses an open-ended timeline and scope of the project whereas we have a plan for the tasks that need to be completed and when.

To handle version control and sprint planning, we can leverage resources such as Gitlab or Github. Any additions to the code base will be actioned by pull requests, where our team should request 1 review before merging to master to ensure code quality, best practices, and bug prevention.

Gitlab and Github also allow the option to create stories in order to ticket individual steps or features needed to be done to complete our project. An alternative could also be to use Trello, which is more dedicated to creating sprint boards and monitoring progress.

Discord and Text are our two main forms of communication. Quick updates and informal messages are sent through text, while group meetings and project-specific discussions take place within Discord.

3.3. Project Proposed Milestones, Metrics, and Evaluation Criteria

Data Milestones:

Get enough data to properly train the model (# rows/data points > 1000)

Determine appropriate prediction for dataset (Classification, Clustering, Regression, Ranking)

Establish data collection mechanisms (mysql and python)

Quality of data is established (null/invalid values < 5% of total data)

Model milestones:

Machine learning model is created (accuracy > 50%)

Model is refined (accuracy > 80%)

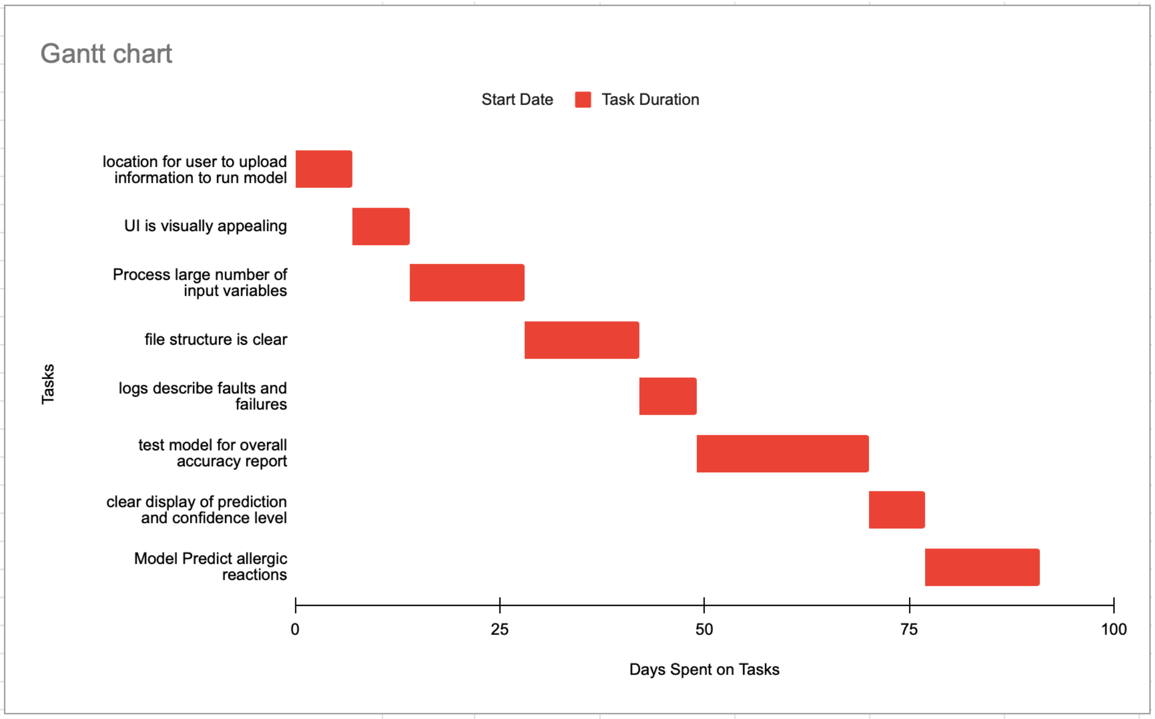
Model is finished (accuracy > 90%)

Model gives a rapid response to the user (time to complete request < 5 seconds)

Model handles a large number of input variables (possible factors in calculations >= 20)

User is able to simply input data through a user interface (less than 5 clicks to complete)

3.4. Project Timeline/Schedule



3.5. Risks And Risk Management/Mitigation

Possible risks and probabilities:

- Model never reaches our minimum accuracy requirements (.7, there could be any number of reasons the model does not meet the goals such as a lack of a common link between the variables, or there is not enough data to properly train the model)
  - Mitigation technique: Compare AI model to others so that if a desired accuracy is never reached, we can explain why or suggest a different iteration for future attempts
- Front-end, backend, and model are unable to connect to each other (.2, even with a simple model, communication between all parts should be feasible without much difficulty)

    - Mitigation technique: Each system should be stress tested individually, but must also demonstrate the ability to effectively communicate with all other involved systems.
- User cannot upload/use their own information to get a result (.3, getting the user to upload information in the correct format may be tricky, but it should be possible)
    - Mitigation technique: We can test the import functionality with our own personal data and include major data importing formats to improve accessibility (such as .csv files)
- Error handling/logging not implemented (.2, most error handling and logging will be built alongside the major components of the project, so the risk of this problem occurring is low)
    - Mitigation technique: Each system should log individually to provide a transparent view of any errors or failed connections to other systems
- Code structure is unclear (.05, any system which divides the front-end, backend, model, and data into separate components will suffice the minimum requirements. Even if this is not followed, the requirement should not impact the model's functionality.)
    - Mitigation technique: Coding structure must be simple and clear, and any formatting should be consistent and listed in the README portion of the project

3.6. Personnel Effort Requirements

Task	Required person-hours	Explanation
Model predict allergic reaction	20	Since none of us were very familiar with machine learning going into the project, we will need to ensure we understand the process along with implementing it.
Process large number of input variables	12	While most of this is done while developing the model, ensuring that the input variables are all being used will take time.
Clear display of prediction and confidence level	3	This requires accessing the model to get these pieces of information along with creating components in the display for these parts.
Location for user to upload information to run model	4	The task is not too complex and should not require much work. Time also includes time to test the upload process and ensure everything works.
UI is visually appealing	2-4	The time we could spend on

		this should vary depending on how much time we have, as it is less important than the functionality of the project.
Collect data to train the model	2-8	The time this takes can vary greatly, as it will be difficult to source legal and ethical data if such data is not provided for us
Completed within 2 semesters	N/A	Since this is a resource constraint, it does not have hours dedicated to it.
Test model for overall accuracy report	30	Testing and retraining the model depending on our needs may take the longest amount of time, as it may involve switching to a new type of model if the one we use does not work.
Logs describe faults and failures	2	In order for us to understand whether the process is happening as it is supposed to, we will likely need logs along the way, but we should spend some time at the end making sure all possible outputs have log output and can be traced or interpreted.
File structure is clear	1-2	This should be planned at the beginning so that the code does not have to change structure much, but should not take much time.
Code is well-documented	2	This should be done as code is created, but allocating time to clean up and explain the repo at the end if there is time would be beneficial.

Table 1: Table detailing each major task, time to complete in hours, and explanation for how we estimated the times.

3.7. Other Resource Requirements

Computing power: since the machine learning model will be a complex model, we may need to utilize additional computing power beyond what we are able to produce ourselves.

Datasets: to train and test our model we will need a large dataset with basic medical information to ensure any factors can be accounted for by the model.

Monetary: if we are not careful, it is easy to incur a large charge when using the cloud platforms to store and run our model, which could affect how often we run and test the model.