Functional Specification

Data Assistance Chatbot

Gareth Hogan 20379616
Jack Farrell 20352136

Table of Contents

1. Introduction

1.1 Overview

The system we are building is called Minerva, Minerva will be a chatbot specialising in interfacing with and helping users interact with databases and database tools. Minerva will be able to help users unfamiliar with database technology and how to effectively utilise data to its full potential. Our system hopes to be able to help people search, graph, visualise and gain insights from datasets.
Our system will be presented in the form of an interactive chat interface, a simple and clean page for users to enter queries and interact back and forth with Minerva. The user-facing portion of our project will comprise the chatbot technology presented to the user as a conversational interface, the chatbot will analyse and interpret user input, sending requests into the backend based on what the user wants to do. In the background will be a variety of database functionality and technology which Minerva can trigger based on user input, to search the data, graph it or run other functions. Our database will also be contained here away from the user, Minerva will be designed to not be database specific, we could deploy this technology with different types of databases based on the users and their needs.

1.2 Business Context

Minerva is designed to be applicable in lots of different use cases and business environments. Anywhere employees (users) have to trawl through or want access to large sets of data or records Minerva could help improve the efficiency and productivity of users interacting with the database. Users would no longer need special training, or lots of time to understand how to search a database properly, It would also allow users who are not familiar with database technologies to be able to retrieve data from the database. They just need to input what they want and Minerva will interpret this and take care of the search, returning the results to the user in a simple and readable format.

1.3 Glossary

Term	Description
API	**Application Programming Interface -** Enables two software components to communicate with one another.
SQL	**Structured query language -** A language that is used to build and search databases.
Natural Language	This refers to text written in language that humans naturally use, it is difficult for computers to understand because of grammar, semantics, conversation quirks and slang

Chatbot	A computer program designed to communicate similarly to a human through a message-like interface
Prompt	The Natural Language statement that a user will type into the interface, is the initial input of the system which goes through processing
Query	An SQL command that returns specific data in the database.
Output	What the system will return to the user
Conversation	A continuous flow of chat with the chatbot, involving multiple prompts from the users and output from the system responding to each prompt

2. General Description

2.1 Product / System Functions

The product is a chatbot interface that allows users who are unfamiliar with database languages to interact with the database without having to spend time and energy learning how to extract the data. The system would be able to understand the user's natural language prompt, process it and respond in turn, if the user has asked for help or entered something Minerva can't do the system can respond with helpful tips and pointers, if the user prompt is correct the system can transform the natural language into a query for the database, process the results and show them back to the user where they can add further prompts if needed, such as asking to visualise data in a graphical format instead of a tabular raw representation.

The system functions as an interactive chat interface, the main operations it goes through are taking in a user query, processing and interpreting it, triggering the correct function based on the query, invoking the function on the database and returning the results back to the user.
Our system is designed to be extendable, working on different databases and also we may implement more functions to work on the database throughout the project. Below you can see the different functions our system will be implementing, they are ordered roughly according to the natural flow we expect a conversation to follow

- System Startup
    - Display Interface
    - Welcome Message
    - Load Database
- User Prompts
    - Take in User Prompt
    - Interpret Prompt
    - Respond to incorrect prompts or help requests
    - Ask User for Missing Info

    - Invoke the correct backend function for a prompt
- Backend Processing
    - Preprocess Prompt (Stem, Reformat, Reduce)
    - Translate to SQL Query
    - Send Query to Database
    - Clean up Data
    - Produce Graph
    - Produce Table
    - Send results back to Interface
- Displaying Results
    - Display the result to the user
    - Explain what system did
    - Take in additional prompts (refine/filter requests to start the cycle again)

2.2 User Characteristics and Objectives

The expected users of our system would differ based on the database(s) that Minerva is installed with at the time, but in general, it is expected, as said above in 1.2 Business Context, that users do not have a lot of knowledge of databases or data search, they would have domain knowledge of their work and the data contained within the data but would find it hard to search through efficiently or use it easily (e.g. A nurse would understand all the data in a patient record database but might struggle to search the thousands of records or display them in a meaningful way).
Users will generally understand what they want and how to understand the results, Minerva will handle how to translate what they want into results.

Below you can find a breakdown of the major use cases or functions that a user may access our system to use. Each major function is broken down into smaller examples of what it can do and examples of what user prompts could ask for

- Search Database
    - Individual Entry Search
    - Group-By Search
    - Overtime Search
    - “Can you show me the data we have for ‘John Doe’”
    - “Can you show me all records that have blood type AB”
    - “Is there any data with these three factors X, Y, Z”
    - “Can I have all the records from the last 72 hours”
- Refine/Filter Data
    - Filter by attribute
    - Filter by time
    - Reduce data
    - Summarise Data
    - “Can you take out all the records where the age is less than 18”
    - “Can you only show me results from last year”
    - “I only need the top 20 results”
    - “What is the average of this column”
- Graph Data

    - “Can you make this a graph”
    - “Can I see this as a line graph”
    - “Are you able to make this a bar chart”
- Help Function
    - Prompt missing information
    - Display Help Message
    - Explain System Uses/Functions

2.3 Operational Scenarios

There are many user scenarios that Minerva could aid users in, it depends on the loaded database as to what the user can do. One of our main examples of a use-case scenario is shown below where a medical patient database is loaded into Minerva and a Nurse or Doctor wants to interact with it.

Medical Example Scenario

In this scenario a patient record database is loaded into Minerva, this database is a huge repository of all patients and their details, names, ages, blood types, injuries, doctors, surgeries, allergies, conditions, etc. This database could contain thousands or hundreds of thousands of patient records, which would make it extremely hard to easily search and manipulate for people without database experience. The users who would want to use this database would be people like Nurses and Doctors, they might want to search for specific patients, compare records, extract records by blood type or condition, or visualise the records of different patients with similar symptoms. Minerva will help users to easily and quickly get results without training or experience with databases.

Below are some example interactions between Minerva and a User in this scenario:

Retrieving Records that Match a Condition

A Nurse wants to see all patients that belong to the AB-negative blood group, and then filter it further after the first result is returned

1. Open the Minerva chat interface
2. User enters their query into chat “I want to see all patients with AB negative blood”
3. Minerva takes in the query and processes it, querying the database
4. Minerva displays in chat all patients that match the condition in a table format
5. User wants to refine further, and enters in chat “Can you only include the last 6 months”
6. Minerva uses the new condition to further restrain the first result
7. Minerva displays a refined table in the chat

Graphing Attribute over Time

A Doctor wants to see if there is any trend in X symptoms over time

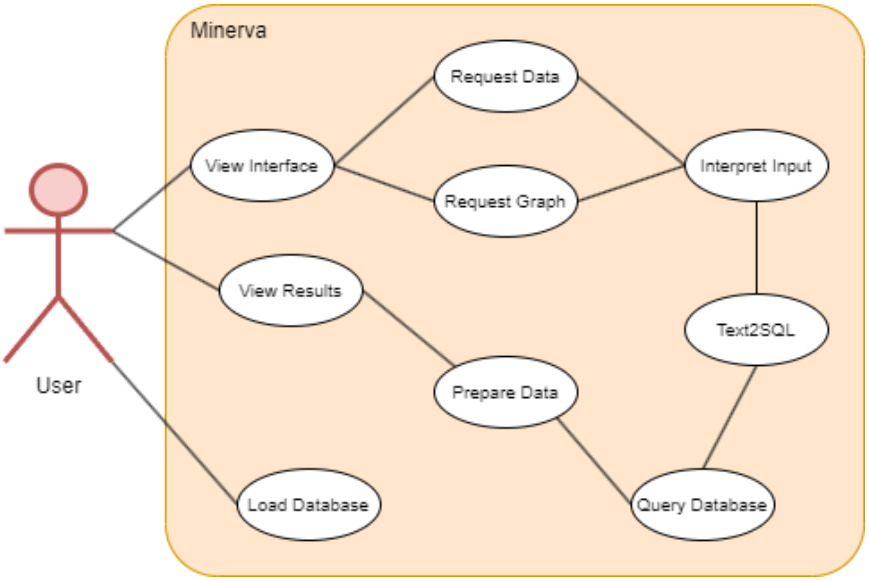
1. Open the Minerva chat interface
2. User enters their query into chat “Can you show me how many patients had a distended bowel in the last 4 weeks”
3. Minerva takes in the query and processes it, querying the database
4. Minerva displays in chat all patients that match the condition in a table format
5. User wants a graph instead of a table, “Can you graph that”
6. Minerva transforms the returned table into a graph over the time specified
7. Minerva displays a graph in the chat

2.4 Constraints

For our system, there are no major constraints placed on the development team or the user, a generic browser will host the interface and the development team will develop the system to perform and respond in reasonable time to the user's queries.
On the backend, the developers will be using the latest versions of the libraries and software required to develop Minerva, including Flask, Python, chatterbot etc.

3. Functional Requirements

In this section, you can see our use case diagram which shows the different things a user might interact with the system to do. In the diagram, the use cases are described at a very high level, in reality, each of the two main use cases contains a lot of different queries and functions that a user can ask for or interact with.



Below you can see the list of our functional requirements for the system.

3.1 - User Input

- **Description -** The system must be able to accept user input through the chat interface in order for users to be able to interact with the system and use it.
- **Criticality -** Without user input the system would not be able to function as it would not be able to evaluate what the user wants, generate the SQL queries nor would it provide value to users.
- **Technical issues -** Creating the chatbot and the rules that go along with it will be a challenge and will take a long time with implementation and testing.
- **Dependencies with other requirements -** User input will be required for the rest of the system but will be directly related to the 3.2 text-to-SQL system

3.2 - Interpret Query

- **Description -** The system must be able to identify what the user query wants and then route it to the correct subsystem, does the user want a graph or a table return or are they asking what Minerva can do.
- **Criticality -** Without interpretation, we cannot know what part of the system the user wants to use and won't trigger the correct subsystem
- **Technical issues -** We need to pre-empt what potential queries users may have and then transform/train these into chatbot rules that direct to the correct subsystems
- **Dependencies with other requirements -** This is a direct successor to 3.1 User Input, being always completed before any other successive functions

3.3 - Help System

- **Description -** The system should be able to output what it can do and show users what it is capable of, catching when users try to do things that the system cannot support
- **Criticality -** This step ensures that we can display to users what the system can do in case they keep trying to do things the system is incapable of doing
- **Technical issues -** We need to properly handle user queries that cannot be done, and then redirect users to things the system can do
- **Dependencies with other requirements -** 3.2 Interpret Query will be responsible for first catching queries we cannot complete and directing here

3.4 - Text-to-SQL

- **Description -** Taking a natural language query from the interface and processing it to produce an SQL query interpretation for what the user wants

- **Criticality -** This system is responsible for generating the queries as without it the system would not be able to gather the information from the database to display to the user

- **Technical issues -** Text-to-SQL is an already made system that we would need to implement into our code but it may have errors within it that we would need to change for it to work properly in our system.

- **Dependencies with other requirements -** Text-to-SQL is dependent on the input received from 3.1 User Input and is required for the rest of the system but directly required for 3.5 SQL API.

3.5 - SQL API

- **Description -** The API will be used to link our chatbot to the database and be able to send queries and receive data, it interacts with the database once we have a query from the user

- **Criticality -** This is our link to the database as without it we would not be able to reach the database to submit our queries or get the data.

- **Technical issues -** We need to make sure that this API is general enough to fit any loaded database, and it must be able to handle errors gracefully.

- **Dependencies with other requirements -** The SQL API is not dependent on anything within our system but is required for 3.4 text-to-SQL to send the query to 3.6 SQL Database.

3.6 - SQL Database

- **Description -** This is our database that stores the data relevant to the user, it is also where our system will be sending the queries and retrieving that data from.

- **Criticality -** The database is important to the system as it is the reason the user is using our system, without the database that the user wants to get the information from the system would not have value to them.

- **Technical issues -** We have used SQL before so the technical issues relevant to this would be more closely related to the API or Queries not working as intended

- **Dependencies with other requirements -** The biggest dependencies on the database is the 3.4 Text-to-SQL creating a query that it can recognize. It is also required for generating anything for 3.9 Output.

3.7 - Graphing Function

- **Description -** When the user wants to visualise the records instead of just the raw data that matches the query, this function must convert the query data into graphs.

- **Criticality -** This function is important to help the usability of the system, converting large amounts of data into helpful graphs for users

- **Technical issues -** We need to have a general function that fits many data formats that could be returned from the database and also supports a few different types of graphs

- **Dependencies with other requirements -** This function depends on the data returned from 3.5 SQL API

3.8 - Database Loading

- **Description -** This is a small part of our project, allowing the system to switch between databases for testing or demonstration purposes

- **Criticality -** This function is critical for the demonstration and testing, users won't be using this function much

- **Technical issues -** This aspect of our project demands we design the system in a generalistic way, making sure not to tie one specific database into the system and always thinking of swapping databases, if we design the system correctly this function should be extremely simple

- **Dependencies with other requirements -** This function does not depend on others but will change the database in 3.6 SQL Database

3.9 - Output

- **Description -** This is the output of the system where the user input is answered and returned to the user via the chatbot

- **Criticality -** We need to show the output otherwise the system will not be able to assist the user.

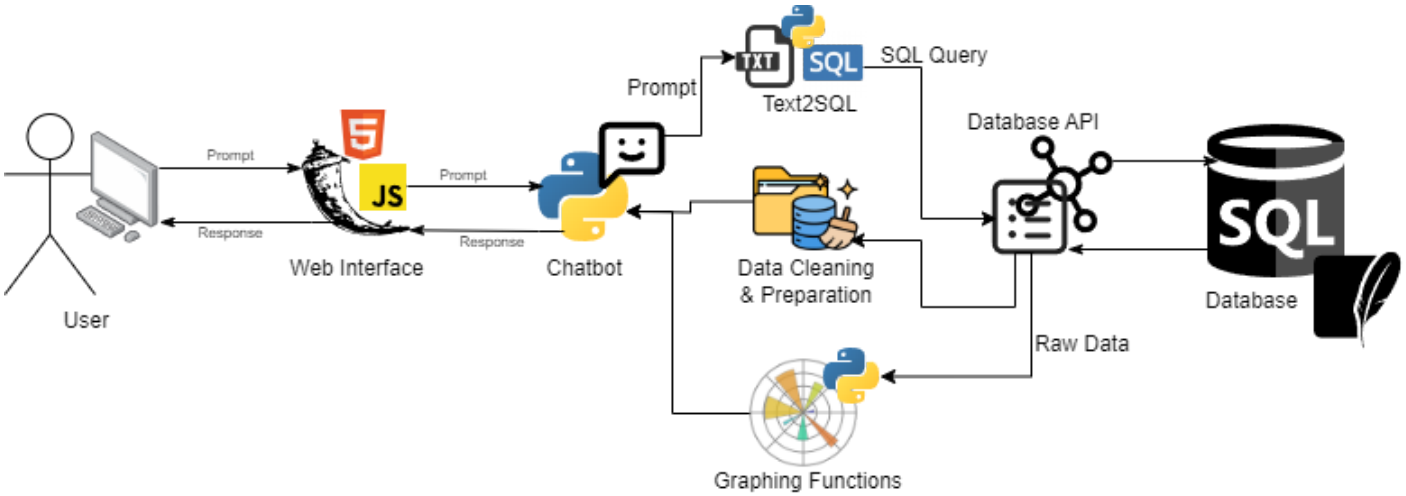
- **Technical issues -** we will be using 3.3 SQL API again to give the query answer back to the user alongside using the chatbot in 3.1 User Input to display the data they requested.

- **Dependencies with other requirements -** The output is dependent on the data that 3.6 SQL Database has sent through 3.5 SQL API.

4. System Architecture

Here you can see the main flow of our system and the major subsystems that will be broken down into, one end is the user-facing interface where the system will interact with users, and down deep at the other end lies the database that is loaded into Minerva
Below you can see our System Architecture Diagram, which breaks down the parts of our system and some of the expected technology we are going to use to develop these subsystems

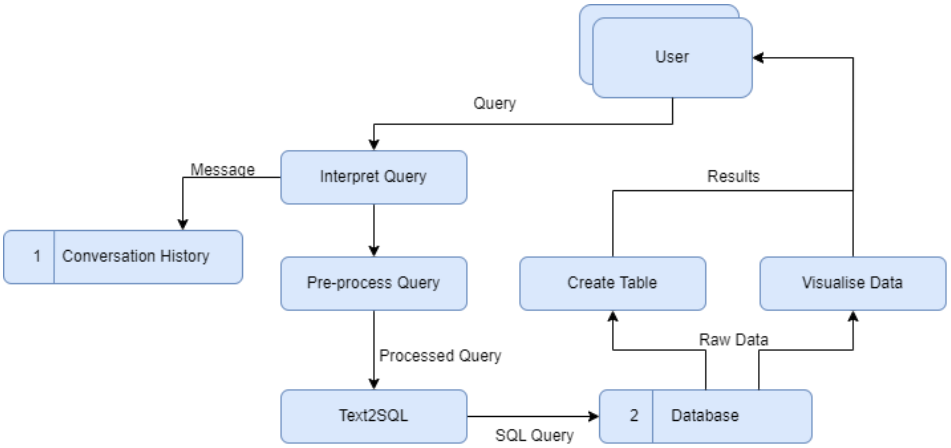
- The user interacts with Minerva through our web interface, hosted on Flask and styled with HTML and JS.
- Also in Flask behind the interface lies our Python chatbot, powered by the chatterbot library, this is where queries will be interpreted and routed into the backend
- The backend API and function area contains many subsystems that deal with all the things our users might want to do. It processes queries into SQL and queries the database, transforming the raw data into results that are displayed back to the user via the chatbot interface



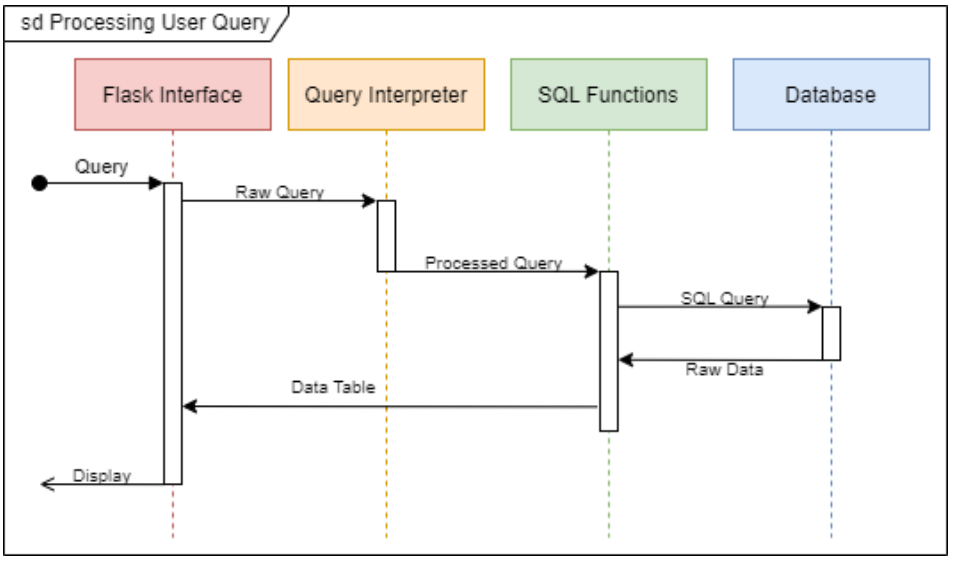
The user will interact with Minerva through the web interface, it is through this interface that the conversation between the user and Minerva takes place, the user enters prompts and Minerva responds, the user may ask for help, want to know what Minerva can do or can enter a prompt to search the database. These prompts are interpreted in the chatbot subsystem, where the correct response or function is selected, the system can then respond directly to the user via the interface or can send the user prompt into the backend to be turned into a query and used to extract data from the database, this data is transformed and returned through the interface to the user. This process of prompts and responses continues until the user is satisfied with the results.

5. High-Level Design

Shown in this section are some diagrams and explanations of the flow of our system, how different parts of the system interact with each other and how the flow of control and data happens to produce the output we want from Minerva. Below you can see our Data Flow Diagram, which models how data will flow through our system and be transformed. In it you can see the flow of the user's query, being interpreted, pre-processed (stemmed, stop word removal), and then translated in SQL to query the database, the data returned then makes its way back to the user either transformed into a table view or potentially visualised in a graph. In our diagram you can also see we are going to probably need a conversation history store, this will enable Minerva to have more sophisticated multi-prompt conversations where the user might refine their request more.



Next, you can see a subset of our system flow in the sequence diagram below, this again shows the flow of our system but now in the context of what part of our system does what and in what order. The user will always be interacting with our flask interface so this is where the query always begins, this is then sent into our chatbot and interpreted and processed, routing it into the correct backend function where it will be translated into a database SQL query and raw data retrieved, this raw data returns into the backend to be cleaned up and made into a presentable format for the user, in this case a Table of the data found. This Table is then presented to the user back through the flask interface.



6. Preliminary Schedule

Our first priority is to get a lot of research and prototyping done, along with discussing the design of our system before we jump into development. The nature of our system means the design is very important, we want to get it right the first time and not have to hack systems together or go back and redevelop systems that we did badly the first time.

For functionality of what the user can do, we are following a hierarchy of what we want to get done, we want to get the search functionality working first as this is the base of our system, converting user queries into SQL and retrieving data from the database.
From then we can expand the functionality of our system to use this retrieved data in more complex systems like visualisation and graphing, we do not want to take on too much work at once so we will work in stages, getting one thing working before trying more ambitious ideas.

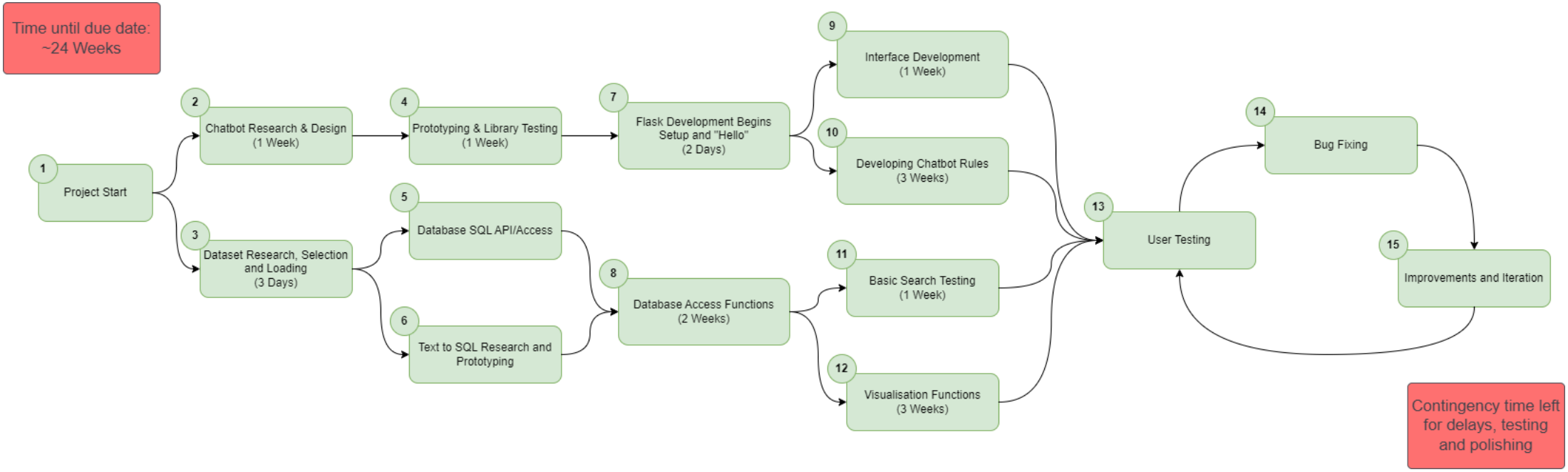
To get the functionality working we will need to design and develop the basic flow of our system early on, getting the basics working and then iterating and improving. The chatbot will be developed rule by rule, first training it on the basics, and getting the necessary ones working well and then we can expand it rule by rule, allowing it to recognise and respond to more user queries. At the same time, we will be developing the back-end database interaction functions, conversion of queries to SQL and database interaction functions.

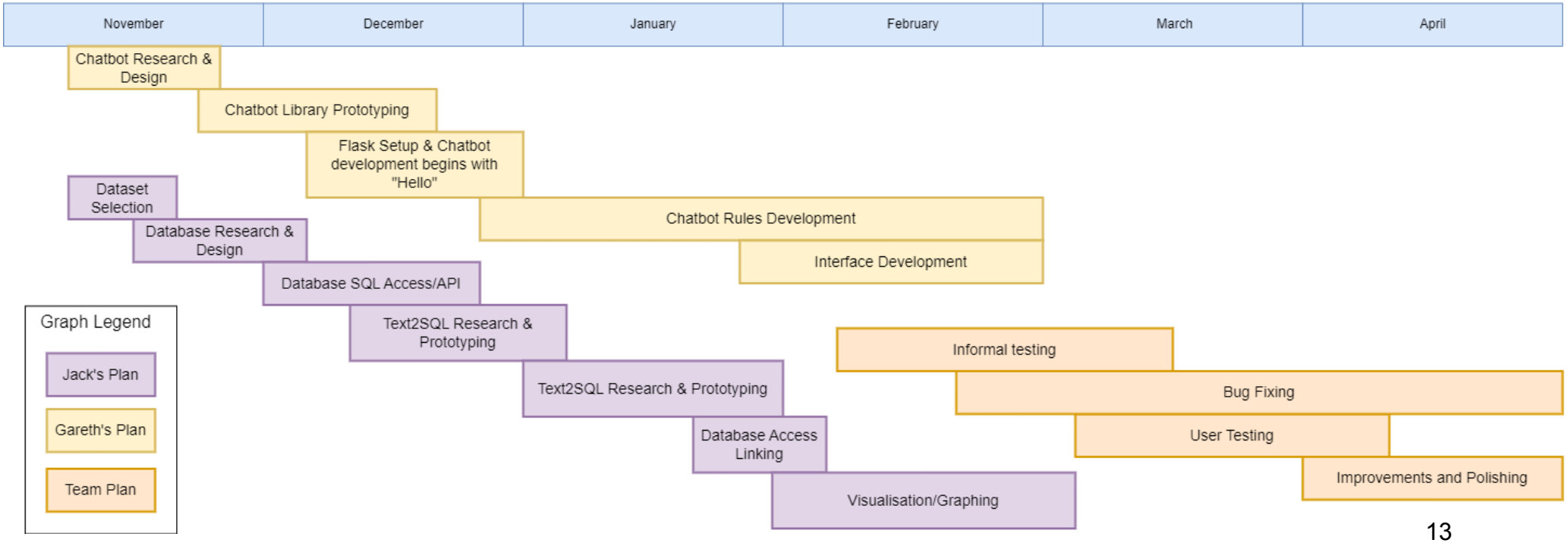
With this approach, one of us will work from the front inwards, starting with the chatbot interface and then linking in the backend database. The other person will start at the database, getting it loaded and working before working outwards to link up with the front end.
Once we get the basics up and running, we can work together to go over, improve and expand the system. We want to design our system to be general and expandable so that we can easily add more rules to the chatbot, and add more functionality to the database, this method will help make our development much more efficient.

Once the system is mostly up and running we can then start looking at testing and verification. This step is extremely important to us, it will help to get outside views of our system, two of us won't be able to think of every possibility of query that a user may input, so our user testing will not only help verify our system is working correctly and fix bugs, it will also help us expand and improve the functionality for users.

As we move towards the end of the project and the deadline approaches we are going to finalise the features we want to complete in the system and stop the development of new features or rules. This way we can move our focus to polishing and testing, making sure all the components are up to scratch and bug-free and testing all the flows that users can go down through user testing and feedback.

On the next page you can view our preliminary plans for development in a PERT and Gantt chart, they break down what order we plan to develop the system in and also who does what. Towards the end of both, you see we want to try leave a lot of contingency time for testing and refinement of the system where we will work closely together to polish off the system.


Time until due date:
~24 Weeks
1
Project Start
2
Chatbot Research & Design
(1 Week)
3
Dataset Research, Selection and Loading
(3 Days)
4
Prototyping & Library Testing
(1 Week)
5
Database SQL API/Access
6
Text to SQL Research and Prototyping
7
Flask Development Begins Setup and "Hello"
(2 Days)
8
Database Access Functions
(2 Weeks)
9
Interface Development
(1 Week)
10
Developing Chatbot Rules
(3 Weeks)
11
Basic Search Testing
(1 Week)
12
Visualisation Functions
(3 Weeks)
13
User Testing
14
Bug Fixing
15
Improvements and Iteration
Contingency time left for delays, testing and polishing


November
December
January
February
March
April
Chatbot Research & Design
Chatbot Library Prototyping
Flask Setup & Chatbot development begins with "Hello"
Dataset Selection
Database Research & Design
Chatbot Rules Development
Interface Development
Database SQL Access/API
Text2SQL Research & Prototyping
Text2SQL Research & Prototyping
Database Access Linking
Visualisation/Graphing
Informal testing
Bug Fixing
User Testing
Improvements and Polishing
Graph Legend
Jack's Plan
Gareth's Plan
Team Plan

7. Appendices

Here you can find links to some of the things discussed in this document and the software we are researching and intend to use:

- Chatbot Library for Python
  https://chatterbot.readthedocs.io/en/stable/
- Types of Chatbots
  https://www.ibm.com/blog/chatbot-types/
- Creating a simple chatbot in Python
  https://realpython.com/build-a-chatbot-python-chatterbot/
- Rule-Based Chatbot Cheatsheet
  https://www.codecademy.com/learn/rule-based-chatbots/modules/rule-based-chatbots/cheatsheet
- Developing a Chatbot with Python and Chatterbot
  https://www.codemotion.com/magazine/ai-ml/develop-chatbot-with-python-and-chatterbot/
- Text-to-SQL
  https://paperswithcode.com/paper/graphix-t5-mixing-pre-trained-transformers
- SQL API
  https://docs.snowflake.com/en/developer-guide/sql-api/intro