Development Plan
Software Engineering

Team 14, Team Name
Aamina Hussain
David Morontini
Anika Peer
Deep Raj
Alan Scott

Table 1: Revision History

Date	Developer(s)	Change
09/22/23	David	Added first draft for tech and coding standard
09/23/23	Deep	Added first draft for proof of concept plan and project schedule
09/23/23	Anika	Added intro and sections 1 - 4 of Development Plan
...	...	...

The following document outlines the development plan for the TBD system. As the scope of the project is further elborated upon, this document will be revised in order to reflect the most updated plan for the project.

1 Team Meeting Plan

Team meetings will occur at least once a week, during the scheduled tutorial time Mondays from 2:30 PM - 4:30 PM ET. All members are expected to attend these meetings, that will be referred to as development (dev) team syncs. If a member has a timing conflict, they must notify the rest of the team as soon as possible, in order to allow for recheduling. Any additional dev team syncs will occur based on member schedule and are not mandatory unless otherwise specified.

On a biweekly basis, the team will meet with the project supervisors to discuss progress and gain clarity on any roadblocks, this is called the stakeholder sync. The timing for these meetings depends on the avialability of the project supervisors and can be subject to change. Although attendance for the stakeholder sync is not mandatory, it is highly encouraged. As with the dev sync, team members should notify the team of any scheduling conflicts although rescheduling may not be possible due to the busy schedules of project supervisors.

Depending on team availability, meetings may occur on campus or virtually. Given the travel time for certain team members, virtual meetings are preffered.

AGENDAS:

Where agendas are concerned, all team members are expected to contribute to the agenda prior to the meeting, and should expect to lead the topics they contribute to the agenda. Any concerns, questions, updates or roadblocks should be jotted in point form on the agenda and can be further discussed in meeting. Furthermore, team members should include a time estimate on agenda items in order to accurately book time for the meeting. Finally, all agendas will be tracked as issues in the project repository and will be closed after meetings.

MEETING ROLES:

Chair: This individual will change every meeting.
They ensure that the meeting stays on track and that all agenda items are addressed. They are also responsible for ensuring that the meeting progresses at a decent pace and starts and finishes on time.

Note Taker: This individual will change every meeting.
They are responsible for taking notes during the meeting and ensuring that all new action items have been recorded and have been assigned to a member. They will also cross off completed action items from previous meetings.

Jira Manager: Anika.
While this role will be elaborated upon in the "Team Member Roles" section, the Jira manager is responsible for updating the Jira in accordance with (applicable) new action items as well as any new information about current Jira issues. They will also be responsible for ensuring that all action items are assigned to a member and have a due date.

2 Team Communication Plan

In order to facilitate communication between all team members, the following tools will be used:

- **Git Issues**
  These will be used to track agendas as well as meeting notes. Git issues provide a centralized location for all meeting information and allow for ease of maintenance as well as updates. Generally, Git issues should not be used as a primary form of communication. Comments under issues MAY be addressed, but team members should expect to discuss via chat or meeting instead. For further details on agendas please see the "Team Meeting Plan" section.

- **Teams**
  Teams will be used for all virtual meetings and as the primary method of (chat-based) communication Meetings should be scheduled ahead of time on Teams in order to appear on all members' Teams calendars. Syncs are scheduled as reccuring meetings on Teams, but can be cancelled or rescheduled as needed. Meetings should only be used for longer updates and discussions, and should not be used for quick updates or questions. The team will use the Teams chat for quicker updates and minor road-blocks or questions. For further details on meetings please see the "Team Meeting Plan" section.

- **Google Calendar**
  The team will use a shared Google calendar which will be updated any time a new deadline, meeting, or action is set. Updates to the calendar are the responsibility of all team members and shall be managed diligently.

3 Team Member Roles

4 Workflow Plan

- How will you be using git, including branches, pull request, etc.?
- How will you be managing issues, including template issues, issue classificaiton, etc.?

5 Proof of Concept Demonstration Plan

The main risks for the success of this project are; none of the members of the team have deployed a web app from scratch before, and that the data provided by the clinical is often in natural language. Since none of the team members have deployed a web app from scratch this may be a risk as we are not certain of all the steps which must be taken to publish a web app. The data provided by the clinical trials being in natural language also is a risk as we will have to research and use a natural language processor to extract the relevant data points.

Therfore to demonstate that we can overcome these risks our proof of concept demonstations will have to both be deployed and able to be ascessed from a device that is not running the source code, and the application must be able to extract the requirements for the clinical trial from the data source. Also the application should be able to display the retreived requiremnts in a GUI of some fashion.

6 Technology

As a result of our system being a pretty complex full-stack application, it will be built using several different tools and technologies, across both the frontend and the backend.

Frontend

- Programming language - Typescript
- Unit testing framework - Jest
- Frontend web framework - React

Backend

- Programming language - Python
- Linter - Flake8, Mypy, autoflake
- Unit testing framework - pytest
- Backend web framework(s) - Django, FastAPI

In addition to the technologies mentioned above, we will be using a few tools that will help us develop our app more efficiently and effectively. First, we will containerize our app by using docker, which will make it easier to test and deploy our application (also use of docker desktop for running/testing our application locally). Furthermore, for our continous integration pipeline, we will be using github actions to run automated tests and linting for both the frontend and backend services/code, and to build our docker images. Finally, we plan to use firebase as our cloud environment, which is where our application will actually run.

There are also a few tools/technologies that still need to be decided upon. We will need to have some database to store client information. It is likely that this will be a relational database, and will be one of Postgres, MySQL, or SQLite. Additionally, it is possible that we will need to use some existing library/tool for parsing trial eligibility criteria, however this will become more clear in the following weeks (i.e., an advanced parsing library, or existing NLP designed for healthcare systems).

7 Coding Standard

For all of our python code/services, we will be following the Pep8 coding standard, and we will be enforcing this with the help of the flake8 linter. We will also be using mypy, which will enforce strict typing (which is usually not present/mandatory in python), meaning all of our function parameters, return types, class variables, etc.. will need to have its type defined. Finally, we will have docstrings present in each module, to give a brief description of what the main purpose of the module is.

For the frontend code we will not be following any formal coding standard, but we will be following best practices when coding in typescript and when using React. Some of these best practices include -

- use functional components
- don't use inline-styles
- maintain a proper import structure (third-party imports first –¿ internal imports below)

8 Project Scheduling

- September 25 - Problem Statement, Proof of Concept Plan, Development Plan
- October 4 - Requirements Document
- October 20 - Hazard Analysis
- November 3 - Verification and Validation Plan
- November 13-24 - Proof of Concept Demo
- January 17 - Design Documentation
- February 5-16 - Revision 0 Presentation
- March 6 - Verification and Validation Report
- March 18-29 - Final Demonstation
- April 4 - Final Documentation
- April 9 - Capstone Expo