Development Plan
Chest Scan

Team 16, Ace
Harrison Chiu
Hamza Issa
Ahmad Hamadi
Jared Paul
Gurnoor Bal

Table 1: Revision History

Date	**Developer(s)**	**Change**
Date1	Name(s)	Description of changes
23/09/2024	Ahmad Hamadi	Completed proof of concept demonstration plan
23/09/2024	Ahmad Hamadi	Completed Appexdix
23/09/2024	Harrison Chiu	Add section 1-8, 10-11

In this document, it describes the necessary technical and social aspects that are needed to effectively create the project Chest Scan. This includes team meeting plans, team communication plans, and team roles. Implicitly, this is a social contract of which each team member agrees and will do their best to meet up to the expectations outlined. All other sections describe the process of the project in a more technical aspect.

1 Confidential Information

The project itself does not use any real patient data, so no confidential information. The training of the neural network will only comprise of publicly available images from open-source datasets. If real patient data is to be used in the future, we will ask the necessary permission to use it and that data will be anonymized. We must also comply with all related medical privacy laws.

2 IP to Protect

The project uses technologies such as Python, Pytorch, and React. All our team members agree to their respective license and terms of agreement.

- **Python:** https://docs.python.org/3/license.html#psf-license
- **Pytorch:** https://github.com/pytorch/pytorch/blob/main/LICENSE
- **React:** https://github.com/facebook/react/blob/main/LICENSE

Acknowledgement of Terms
As a member of team working on this project, I acknowledge that I have fully read, understand, and am in full compliance of the above licenses and terms of agreement.
Signature: HC HI GB AH JP
Date: 23 September 2024

3 Copyright License

The project will use the MIT License. It is a permissive open-source software license that allows anyone to use, modify, and distribute the software with minimal restrictions, provided the original copyright notice and license are included.

The project's license is found here: https://github.com/harrisonchiu/xray/blob/main/LICENSE

4 Team Meeting Plan

Our team will have virtual meetings on Discord weekly. We will discuss our upcoming plans, our progress, and any blockers. We may have impromtu meetings if we need to discuss more things or if we have an unexpected development in the project. Once a month, we plan to meet in person on McMaster Campus. This is to physically share any development we may have. Collaborative help is easier in person. We plan to also meet with our supervisor once a week on Microsoft Teams. The meeting is to update him on our progress, plan, and any blockers. Gurnoor will lead our discussion. We may ask him on advice or any plans he may have for us and the project.

All meetings are structured such that we have a specific topic we must discuss (e.g. what languages do you want to use?). Gurnoor will be the chair for all meetings and will lead the discussions. He will also send emails to our supervisor as needed. Hamza will take notes of our meetings.

5 Team Communication Plan

Our team will communicate through Discord. It will be used to host our weekly virtual meetings and to text any information to each other. Discord is a great platform as it logs all messages, is easy to use, can share files, and can easily set up meetings. That way, everyone in the team is up to date with any development.

Github is our remote repository to host all our code and documents. It also has a kaban board and issue tracking. Our team will assign issues to different members and track its progress. In this way, everyone can see everyone's progress on their assigned task. We can also see our general progress and how far we are from completion.

6 Team Member Roles

- Harrison Chiu
  - **Repository Admin**. Responsible for ensuring the code base organized and setting up any git-related aspect such as the CI pipelines or fixing branches.
  - **Scrum Master**. Responsible for creating and updating Github issues based on progress or any new information.
  - **Developer**.
  - **Reviewer**.
- Hamza Issa
  - **Note Taker**. Responsible for taking notes during the meetings. They may use these notes to create email drafts to update our supervisor if necessary.

  - **Developer**. Expert in machine learning. Responsible for researching neural networks to fulfill our problem statement.
  - **Reviewer**.

- Gurnoor Bal
  - **Liason**. Main communicator between our team and any third-party which includes our supervisor. Responsible for sending the finalized email draft. Also responsible for leading any discussions in meetings to ensure we follow our agenda.
  - **Developer**.
  - **Reviewer**.

- Ahmad Hamadi
  - **Developer**. Expert in machine learning. Responsible for leading the research and development in machine learning to create the best neural network.
  - **Reviewer**.

- Jared Paul
  - **Developer**. Expert in full-stack development. Responsible for leading the development in creating the full-stack app that runs the neural network. Also responsible for helping train the model.
  - **Reviewer**.

7 Workflow Plan

7.1 Version Control

Git is used as the version control and Github is used to host our repository.

The root branch is `main`. All developers will branch off the `main` branch to work on issues. Once the work is complete, a pull request is made to merge the changes into `main`. It must be reviewed and approved by at least one other team member. The pull request must also pass any CI tests before it is merged. CI tests include format and build checkers. The `main` branch is protected and no one will directly make changes to it.

7.2 Issues

Github Issues and its Kaban board is used to track progress and assign each member tasks. Each issue's topic focuses on one general feature. For example, "Development Plan" or "Clean training data" are 2 issues. If "Development Plan" is too large, then it should be split up into sections (e.g. "Development

Plan Section 7-11"). One or more members may be assigned to one issue depending on the scope and complexity. This will be discussed in our meetings. Each issue has a short description explaining what must be done and why (if needed). New issues are created as necessary either whenever a new milestone is assigned, when something must be fixed, or when a member feels like making a new change.

Each ticket is in one of four states:

- **To Do:** Tickets that must be done soon but have either not been assigned yet or have not been started.
- **In Progress:** Tickets that have been assigned and the assignee has started working.
- **In Review:** Tickets that are completed and have made a pull request and are waiting for verification.
- **Completed:** Tickets that are completed and have been merged into `main` branch.

7.3 CI/CD

Github actions will be used to handle any continuous integration and deployment actions. Linters and formatters will be used to ensure stylistic standards. It will also auto build certain files to ensure the changes work. These actions are only done on pull requests trying to merge into the `main` branch. All pull requests must pass all CI tests in order to be merged.

7.4 General Workflow Process

General workflow from beginning of a task to the end involves the aforementioned version control, Github issues, and CI workflow. The process is generally as follows:

1. A new issue is created. It details some feature that must be developed. It is in the `To Do` status and includes:
    - A short description detailing what must be done (and why if relevant).
    - Assignee(s) that will work on the issue.
2. The local repository of the assignee any changes from the `main` branch.
3. The assignee creates a new branch off of `main` following the branch naming convention detailed in section `Coding Standard`.
4. The assignee makes the appropriate changes to fulfill their task. They may also:

- Write new unit tests on their newly created functions
- Document their changes
- Build any relevant files

5. The changes are pushed and a pull request to merge into `main` is made.
6. CI pipeline runs to check any linting, testing, or formatting errors.
7. At least one other team member reviews the pull request. If they approve it, the pull request is merged.

8 Project Decomposition and Scheduling

The Github repository will contain everything related to our project such as our source code. It is found here: https://github.com/harrisonchiu/xray

The project's schedule is as follows:

- **Sept 24** Problem Statement, POC Plan, Development Plan
- **Oct 4** Requirements Document Revision 0
- **Oct 20** Hazard Analysis 0
- **Nov 3** V&V Plan Revision 0
- **Nov 10** Design of the Neural network model
- **Nov 13-24** Proof of Concept Demonstration
- **Dec 15** All data is cleansed
- **Dec 27** First draft of neural network is trained and analyzed
- **Jan 10** Good draft of neural network is trained and analyzed
- **Jan 17** Design Document Revision 0
- **Jan 30** Good neural network is trained
- **Feb 5-16** Revision 0 Demonstration
- **Feb 20** Full stack application using neural network is finished
- **Mar 6** V&V Report Revision 0
- **Mar 18-29** Final Demonstration (Revision 1)
- **Apr TBD** EXPO Demonstration
- **Apr 4** Final Demonstration and Final Report

9 Proof of Concept Demonstration Plan

The main functionality required for the success of this project is the accurate analysis of chest X-rays and the generation of corresponding medical reports. The most important part of this process is ensuring that the convolutional neural network (CNN) can correctly identify and extract the key features from the X-ray images. In addition to recognizing patterns in the images, it is also important that the model can generate clear and accurate text that describes the findings in a way that medical professionals can understand. Since this is the core functionality of the project, the inability for our neural network to correctly identify diseases represent our biggest risk.

Our concern is that the neural network has a low accuracy (¡80%) in correctly identifying diseases. If our project cannot identify, then it is unreliable as a tool for radiologists, and thus, potentially useless. Furthermore, it could even misdiagnose with false negatives and prove to be detrimental to patients.

To show that this risk can be overcome, the Proof of Concept Demo will involve manually setting up the backend pipeline for the image analysis and report generation. We will input a set of chest X-rays and run them through the different components of the model to show that it can correctly analyze the images and produce useful descriptions of the findings. The goal of the demo is to demonstrate that the system can accurately process chest X-rays and generate meaningful outputs that could be used in a medical setting.

Another large risk our project faces is the possibility of insufficient testing data. Finding a large amount of chest x-ray images that are labeled by radiologists' reports that are varied enough to train a model can be very challenging. This could be because the datasets are not open sourced or because the available datasets are too small. It directly affects our model's performance. To mitigate this, our team will read research papers on chest x-rays to search for datasets other than MIMIC-CXR.

A large risk involving the web aspect of the project is the possibility that the neural network is too large and too slow to run efficiently on the backend. It may have high latency classifying the images which would affect its usability, especially with multiple users and requests. A very large model would also make it difficult to store in the backend cloud. This will negatively affect continuous deployment speed and scalability. To mitigate both of this, we will scale down images, so they take up less space. The processing will be done in parallel with multiple images. This will reduce the high latency of running the model. Furthermore, the model will be partitioned into distributed nodes, so parts of the model can be run in parallel and be stored separately at a smaller size.

10 Expected Technology

10.1 Languages

- **Python**: It will be used to cleanse any training data, train the neural networks, and run the neural network in the backend of the full stack application. It was chosen for its vast amount of machine learning and data processing libraries. It has a lot of support for machine learning support.
- **JavaScript**: Used to create the frontend of the application. It will create the UI. Chosen because it is very popular to create frontend web applications. There are many online support for creating JavaScript front end and many frameworks to help with it.

10.2 Libraries

- **React**: Very popular frontend UI framework for developing dynamic sites. There is a lot of online documentation. There are many already-made components to help us get started.
- **NodeJS**: Popular JavaScript library to handle backend requests. Lots of support. Old framework, so it is tested and has a large online backing. It also means it is stable.
- **Pytorch**: Extremely popular machine learning library. Will be used to train the neural network. Chosen because nearly all machine learning training in Python uses Pytorch. It is good to stick with the convention so other people can easily understand what we are doing. Also, there is a lot of support for Pytorch.
- **OpenCV**: Image processing Python library. Used to cleanse training data and help test the neural network to check its performance. Chosen because it is an extremely popular image library. It is old so there is lots of documentation, huge community, and has little bugs.

10.3 Linters and Formatters

All linters and formatters are checked through the CI pipeline. All linters and formatters were chosen because they are very popular. As a result, it is known that they have very little bugs, large community support, and the formatters stick to the global conventional style.

- **Flake8**: Ensures Python code aligns with PEP8 style guide.
- **Black**: Ensures Python code is formatted nicely and readable, following the PEP8 style guide. Black has a very strict style and it will make all the code stylistically consistent.

- **isort**: Ensures Python import statements are organized in a consistent manner.
- **Prettier**: Ensures JavaScript code is formatted nicely.
- **ESLint**: Checks for static errors in JavaScript.

10.4 Unit Testing

- **Pytest**: Writes and automates tests to ensure Python code works. All created Unit tests will be kept for future regression testing. Popular testing framework, so our team members already know how to use it.

10.5 Continuous Integration

- **Github Actions**: Github Actions will provide all the tools to set up a CI pipeline. It will be used to check linting and formatting. Used because it is already part of Github. There is no need to create an external CI pipeline.
- **Branch protection**: The `main` branch is protected. Make a pull request to add commits to `main`. This forces developers to create branch and enforce a linear main branch which makes reverting and history tracking easier.

10.6 Other Software Tools

- **git**: The only version control that will be used. This ensures consistency and simplicity in the technical workflow. Github only uses git anyways. Only one repository is created and Github will host it. The project's scope is not wide enough to warrant multiple repositories.
- **Latex**: Used to create and format our reports and capstone documents.

10.7 Pre-trained models

- **ResNet and Inception**: Subject to change as we continue researching. For now, these are chosen because of their popularity and are known to be decently well-trained to localize objects. We may or may not train our own as we learn more.

11 Coding Standard

11.1 Version Control

The naming convention is for branches is `<name>/<short-description-of-change>`. Developers may have a couple nested branch folders after their name. Branch

names must be written in `kebab-case`. In essense, only lower case letters and numbers seperated by hyphens. For example, some good branch names are `harrison/clean-training-data` and `harrison/docs/srs`. Ideally, each git branch is not littered with dozens of sloppy commits and are squashed appropriately.

Git commit messages must be detailed and concise. They must also be grammatically correct, have correct spelling, and proper capitalization. Each commit must contain one specific change and not span multiple (i.e. high cohesion). Nonetheless, they should not be too frequent and atomic which can lead to an unnecessary mess of commits. In that case, they should be squashed. It is up to each member to find the balance.

11.2 Language Standards

Latex: Lines will not surpass 120 character. New lines will break text into seperate lines. This is so text can easily be read in viewers without word wrap.
Python: All Python code will follow the Black code style (https://black.readthedocs.io/en/stable/) which aligns with the PEP 8 style guide but it is more strict. The Black formatter (in the CI pipeline) will ensure all code follows the style. This style was chosen because it is strict and ensures that all parts of the code follows exactly that style. The style mentioned in PEP 8 is too loose and generally results in many different styles. For example, it does not even standardize whether to use single quotes or double quotes for strings. Black is very opinionated, so it enforces all those ambiguities in the styles. Python's import will also be alphabetical and organized into sections by type. The utility `isort` (also in the CI pipeline) will ensure compliance.
JavaScript: All JavaScript code will follow Airbnb's JavaScript style guide. JavaScript does not really have a standard style. Thus, this style was chosen because it is more specific and in depth than most other popular ones such as Google's.
See here: https://github.com/airbnb/javascript

Appendix — Reflection

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. Why is it important to create a development plan prior to starting the project?
   A development plan is crucial because it helps the team outline the project's goals, timeline, and key milestones. It provides a structured roadmap that ensures everyone is aligned on the objectives, minimizes risks, and helps to identify any potential bottlenecks early. Additionally, it allows the team to allocate resources efficiently and track progress, which is vital for staying on schedule and meeting deliverables.

2. In your opinion, what are the advantages and disadvantages of using CI/CD?
   The main advantage of using Continuous Integration/Continuous Deployment (CI/CD) is that it automates the testing and deployment process, leading to faster and more reliable releases. CI/CD ensures that code is regularly integrated, tested, and deployed, which reduces the risk of errors and makes it easier to catch bugs early. The disadvantage, however, is that setting up CI/CD pipelines can be complex and time-consuming, especially for projects that are not well-structured from the start. There can also be significant resource overhead when maintaining and updating the pipelines

3. What disagreements did your group have in this deliverable, if any, and how did you resolve them?
   Our group had a minor disagreement on the prioritization of certain features during the initial development phase. Some members felt that we should focus more on the user interface, while others believed the core functionality should be developed first. We resolved this by conducting a quick team discussion and agreeing to develop the core features first, followed by an iterative approach to improving the interface as we go along. This allowed us to address both concerns in a balanced manner.

Appendix — Team Charter

[borrows from University of Portland Team Charter —SS]

External Goals

Our team's external goals for this project include gaining valuable experience to talk about in job interviews, aiming for an A+ grade, and showcasing the project at the Capstone EXPO with hopes of receiving recognition or awards.

Attendance

Expectations

The team expects all members to attend meetings on time and stay until the meeting is over unless prior notice is given. Leaving early or missing meetings is discouraged unless necessary and communicated ahead of time.

Acceptable Excuse

Acceptable excuses for missing a meeting include personal emergencies, illness, or prior commitments that were communicated beforehand. Unacceptable excuses include general forgetfulness or poor time management without prior notice.

In Case of Emergency

In the case of an emergency, the team member should notify the group through our communication platform (e.g. Discord or email) as soon as possible. They are also expected to provide an update on their progress and reassign any pending work if necessary.

Accountability and Teamwork

Quality

The team expects members to come to meetings prepared with relevant updates and complete tasks to a high standard. Deliverables should be polished and completed on time to avoid delays in project progress.

Attitude

Team members are expected to maintain a positive and collaborative attitude. Everyone's ideas should be respected, and constructive feedback is encouraged. We will adopt a code of conduct to ensure professional and courteous interactions. In case of conflicts, the team will resolve issues through open communication or bring in a mediator if needed.

Stay on Track

The team will use regular progress check-ins and project management tools (GitHub) to ensure everyone is contributing as expected. Members who perform well will be recognized during meetings. If someone consistently underperforms, we will discuss the issue as a team and consider assigning lighter tasks or escalating to the TA if needed. Failure to meet targets may lead to consequences, such as taking on extra work or contributing to team rewards like bringing snacks to meetings.

Team Building

To build team cohesion, we plan to have occasional informal gatherings, such as virtual coffee breaks or in-person meetups, depending on availability. These activities will help us bond and maintain a positive team dynamic.

Decision Making

Our team will make decisions through a consensus-based approach, ensuring all voices are heard. If consensus cannot be reached, we will vote. In case of disagreements, we will hold a separate discussion to explore all viewpoints and come to a solution that works for the majority.