MAIN PROJECT REPORT

Team: Visionaries

Contributors

Maryam Afsarzai:

- Research into ML subjects.
- test cases
- Automated tests

Muhammad Daud Raza Aamir:

- Jupyter notebook

Murtadha Al Najjar:

- Software design
- Python and database code.
- User stories

Shehroz Nusrat Abbasi:

- Writing the HTML and CSS for 3 pages.

Ayan Ahmed Butt:

- Assisted with User stories
- Documenting the prototype
- Provided Evidence of tools used
- non-functional requirements
- Design of the Report

Key features of the design

The Chest Ray Web App prototype enables remote triage of pneumonia using portable chest X-ray images in field clinics. The app has two primary user interfaces: the Health Worker View and the Expert Clinician View, ensuring an efficient workflow between on-site health workers and remote medical professionals.

- **Health Worker View:**

- Allows health workers to create or access patient records by entering basic details (name, age, symptoms).
- Provides an interface to upload X-ray images taken with a portable device.
- Automatically detects suspected pneumonia cases and updates the patient's record.
- Displays a real-time list of patients with a checkbox system for indicating pneumonia suspicion.

- **Expert Clinician View:**

- Shows a list of suspected pneumonia cases with patient details.
- Allow clinicians to review the X-ray image of each patient.
- Provides a diagnosis confirmation and prescription field, enabling clinicians to suggest appropriate treatments remotely.

- **Workflow and Automation:**

- The web app maintains a real-time patient record that syncs updates between field clinics and remote hospitals.
- Automates the initial pneumonia detection and alerts clinicians for verification.
- Enables urgent referrals when necessary.

This Lo-Fi design ensures quick decision-making, remote medical support, and streamlined patient management in low-resource settings.

User stories

Note: a record is patient details with an x-ray image

Title Health Worker submits a record

Narrative:

As a health worker

I want to submit records

So that I can receive prescriptions for each

Scenario 1:

given a connection exists between me and the hospital

When I submit a record

then I am informed of the automated diagnosis

and the record is received on the other end

Title clinician receives records to prescribe

Narrative:

As a clinician

I want to receive records

So that I can write a prescription for each

Scenario 1:

Given a connection exists between me and the clinic

When I write a prescription for a record

and verify the automated diagnosis

Then it should be received on the other side

----

as a user I want the list to be updated as soon as possible

as a user I want the list to be sorted by importance (time added first, automated diagnosis first)

Non-Functional requirements:

Usability:

Both seasoned clinicians and field health workers should find the web application simple to use. The layout should be straightforward, with buttons and directions that are easy to read. Health professionals should have no trouble entering patient information, uploading X-ray pictures, and receiving findings. If something goes wrong, the app should also give the user clear instructions on what to do next. Skilled medical professionals should be able to quickly analyze X-ray pictures and patient records and make recommendations without needless processes.

Accessibility:

Users with different degrees of digital literacy and disabilities must be able to use the online application. For those who are unable to use a mouse, it should enable keyboard navigation and support screen readers for those who are visually impaired. To serve a variety of user bases in various geographical locations, the interface must be accessible in several languages. The program should also perform well on mobile devices and low-resolution screens to assist field health workers who might not

Availability:

To guarantee that medical professionals and clinicians can always access the Chest Ray web app, it must have a 99.9% uptime target. To enable data to be stored locally in the event that an internet connection is unavailable and to sync automatically upon reestablishing connectivity, the system should be built with offline functionality in mind. Backup plans ought to be in place in case something goes wrong to avoid losing data. The system should be promptly restored in the event of a crash to prevent the loss of X-ray images and patient records. Regular updates ought to be scheduled without seriously interfering with clinicians' and health workers' work.

Performance:

To manage several users at once without experiencing noticeable lag, the web application should be tuned. Rapid analysis is required of the pneumonia detection model, with results produced seconds after a picture is uploaded. To guarantee that patient records are obtained and updated effectively, database queries should be streamlined. Even with low-bandwidth networks, the app should load rapidly, guaranteeing seamless operation in distant field clinics. To find and fix bottlenecks, performance testing should be done on a regular basis. All crucial processes should have reaction times of less than two seconds.

Security:

To protect sensitive patient data, the system must have robust security features. TLS and HTTPS should be used for encryption in all connections. Only authorized users should be able to access patient records thanks to role-based access control and user authentication that requires secure login credentials. Encrypted storage of uploaded X-ray pictures is recommended, and data must adhere to privacy standards in the healthcare industry. Common dangers like SQL injection, cross-site scripting (XSS), and

unauthorized access must be prevented by the system. It is important to perform routine penetration tests and security audits to address any vulnerabilities that can endanger data.

User stories implemented:

(1) Health worker submits a record

The health worker fills in a form which sends a POST request to the same route, which extracts the name, age, symptoms and x-ray then runs the x-ray through the AI algorithm then calls the database_functions file which then adds: date-time, algorithm result, actual result and prescribed (which are null for now), note errors are not handled well since this is a prototype. The list is updated synchronously for both sides.

(2) Clinician receives records to prescribe

When the new entry is received on the clinicians side he simply has to click a prescribe button next to the record that opens a form dialogue with two inputs (prescription and result), the form sends a post request, the app receives the data and changes the designated record adding the prescription and the actual result (yes or no)

(3) Update list synchronously

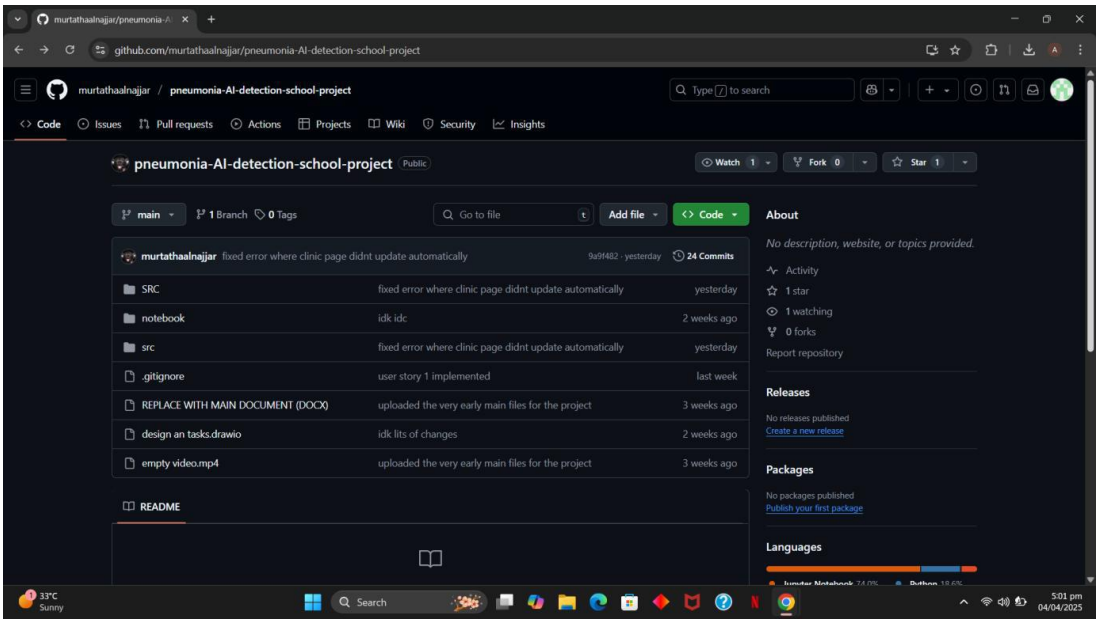
Because web sockets are a new skill to learn, we simply added java-script to both of the pages to constantly send a GET request to the “get_list” route which simply returns json for the list without any security. The java-script is simply a function to fetch the data from the server and render it, which is inserted in a setInterval function to run every two seconds.

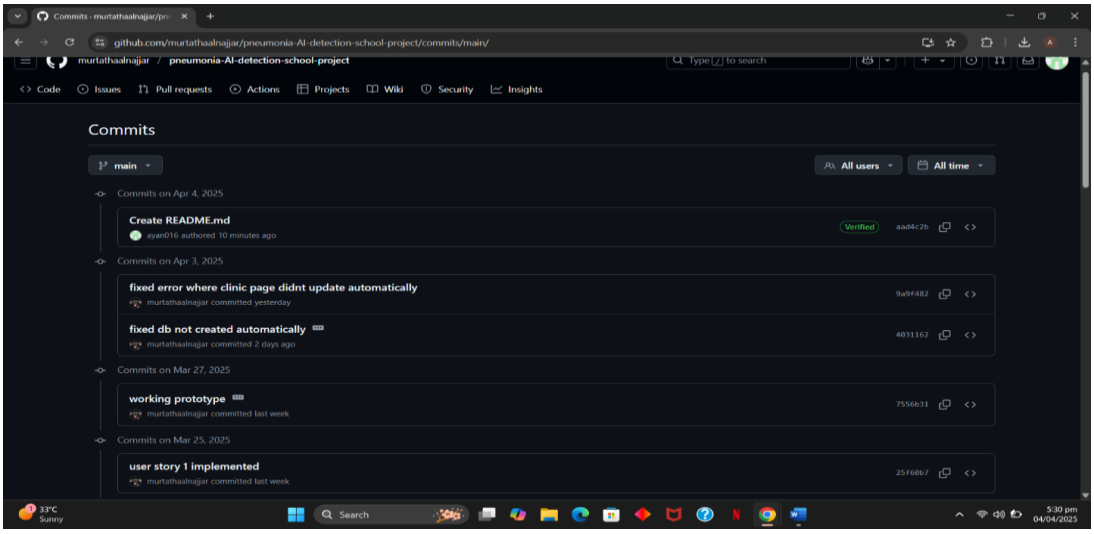
(4) List is sorted by importance

The “Xray_database” class returns the data in order of “actual_result” (records with results from clinicians first) the data is rendered normally for the hospital and in reverse for the clinic.

Evidence of the Tools used

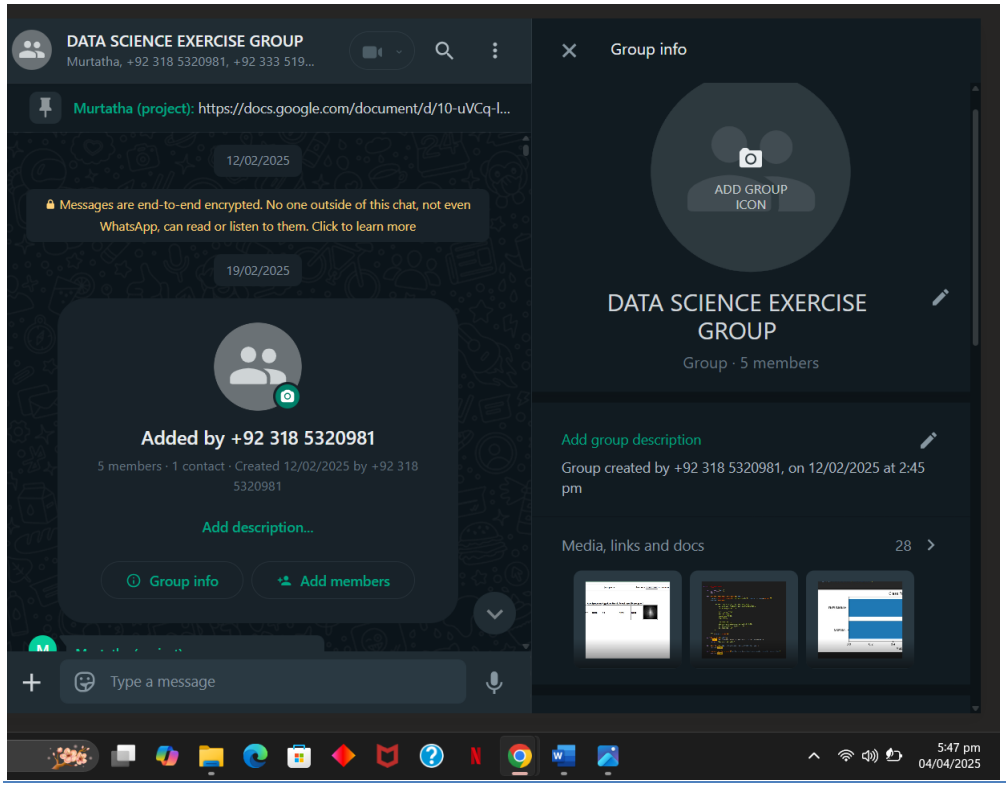
Github:





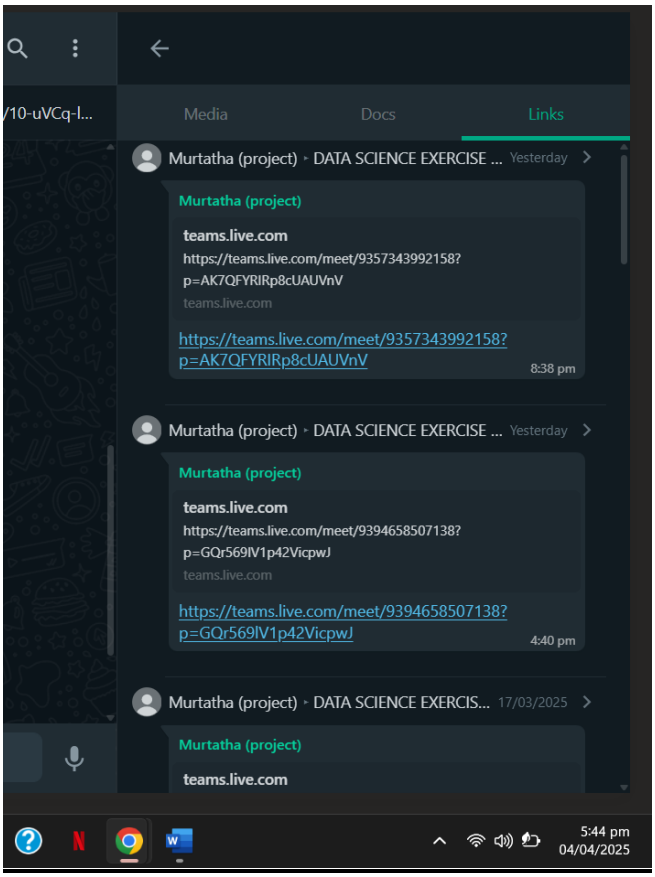
We used a GitHub repository for our project communication and version control. GitHub allowed us to share code, documents, and resources among team members, facilitating remote collaboration. Branching and merging let us work on separate areas of the assignment concurrently without overwriting each other's work.

WhatsApp group:



Our team used WhatsApp for efficient communication, including brainstorming, scheduling, and real-time updates. The group chat function allowed us for easy conversation and sharing of files, links, and screenshots, leading to improved collaboration. WhatsApp enabled effective work delegation, specialization, and collaboration among all team members.

Microsoft Teams:



We also used Microsoft Teams for online meetings to collaborate on our project. We scheduled regular meetings, shared files, assigned tasks using Planner, and communicated through chat and video calls. Teams enabled real-time collaboration, ensuring everyone stayed organized, productive, and on track throughout the project.

Test cases:

	Test Cases	Test Scenario and Objective	Preconditions	Test Steps	Input Data	Expected Result
1	Health worker adds a patient record	Verify that the system correctly adds a patient record through the health worker interface.	Health workers are logged in.	*Navigate the 'Add Patient Record' page. *Fill in the patient details form (Name, Age, Symptoms, X-ray). *Submit the form	Name: "John Doe", Age: "30", Symptoms: "Cough", X-ray: "Uploaded File"	A confirmation message is displayed, and the record is added to the database.
2	Clinician prescribes treatment	Ensure the clinician can add prescriptions to existing patient records.	Clinicians are logged in and patient records are available.	*Select a patient from the list. *Enter prescription details in the provided form. *Submit the prescription.	Prescription: "Antibiotics 500mg", Patient ID: "1".	The prescription is added to the patient's record, and a confirmation message is displayed.
3	Synchronous list update	verify that the patient list updates synchronously across different user interfaces.	Multiple users are logged in on different interfaces.	*Health worker adds a new patient record. *Check the patient list on the clinician's interface immediately	None.	The new patient record appears on all active user interfaces within seconds.

4	List sorting by importance	Confirm that the patient list is sorted correctly by diagnostic importance.	Multiple patient records with varying diagnosis statuses.	*Observe the default sorting order of patient records on clinician's dashboard	None	Patient records are sorted with those awaiting prescriptio n at the top.

Algorithm:

As part of the development process, we created a Jupyter Notebook that contains the full implementation of the pneumonia detection algorithm using a Convolutional Neural Network (CNN). The notebook includes code for data preprocessing, model architecture, training, evaluation, and result visualization.

This notebook served as the foundation for testing the model before integrating it into the web application. While the notebook itself is not deployed, it demonstrates the machine learning pipeline and confirms the feasibility and effectiveness of using AI for detecting pneumonia from chest X-ray images.