AY 2022/23 Semester 1

IFS4205 Information Security Capstone Project Group 3

Tool Assessment Report

Crowd Sensing Medical Application

Oscar Lai Chun Ho	A0204697W
Alicia Tay	A0204813N
Daryl Hung Dejian	A0217127L
Isaac Lai Zile	A0216952B
Lim Jie Rui	A0217086A

1. Project Plan

1.1 Summary

Infectious diseases and resulting pandemics have and will continue to be a serious public health issue, particularly with globalisation quickening the spread. Many health facilities involve waiting rooms, long wait times, as well as counter staff and doctors having to pull out physical paper records on patients. These increase the duration that patients are exposed to such diseases in small, enclosed spaces.

This group is therefore proposing to build a system that manages mass health appointment check-ups during pandemics or virus outbreaks. It will comprise web applications and IoT crowd-sensing devices to be installed in the waiting rooms of the health facility.

Intended security concerns intend to address:

- IoT crowd sensors cannot be forged
- Publicly revealed information is anonymized
- only approved personnel can perform tasks
- Basic web app security

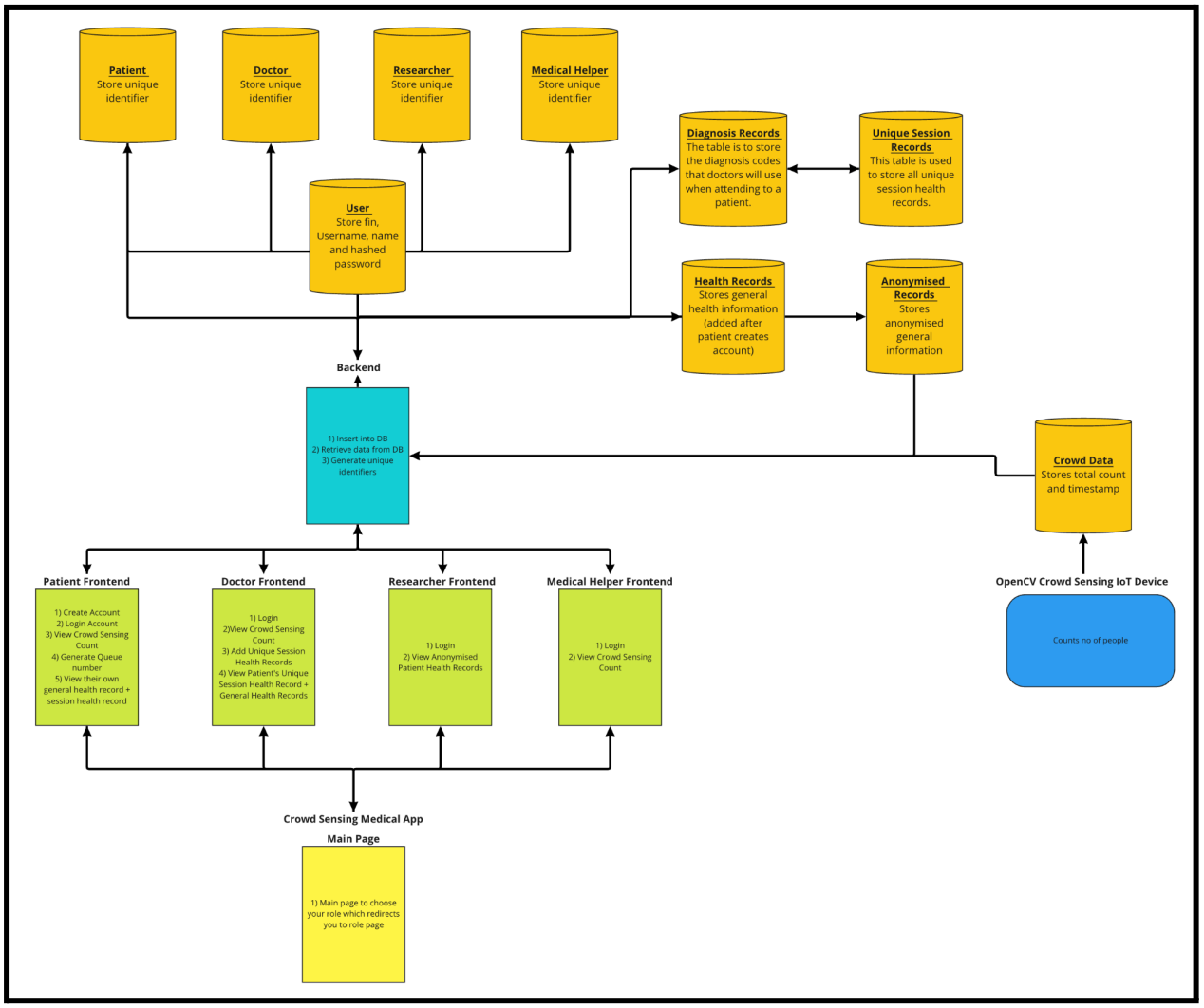

Figure 1: Draft System Design

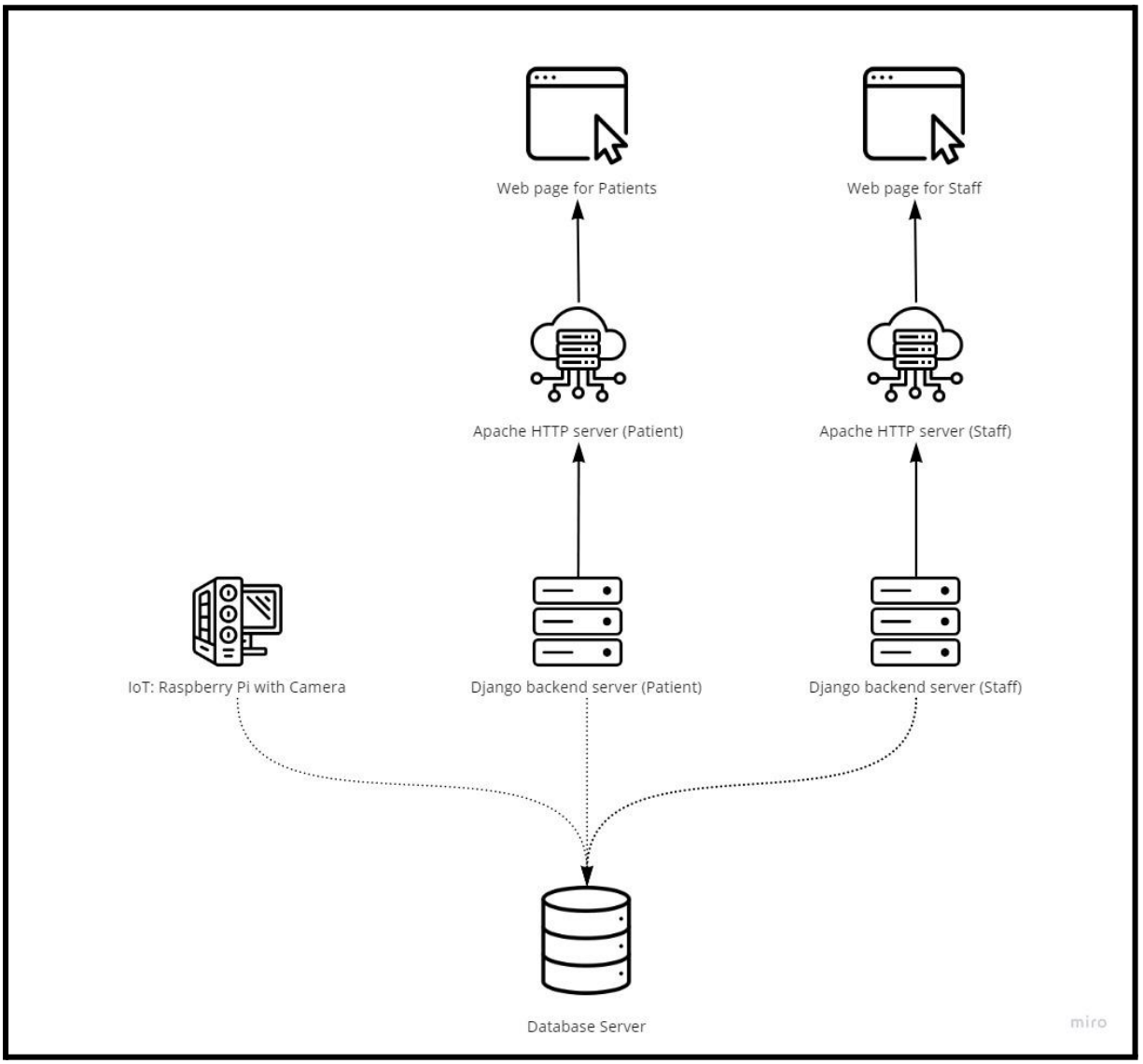

Figure 2: Architecture

1.2 Key Functions of Proposed App

The application will have a main page which will allow you to choose the role. This will allow users to diverge into four different pages depending on their roles: Patient, Doctor, Researcher or Medical Helper page as seen in Figure 1.

The patient's portal enables patients to manage their health records. They can create an account, login, generate a queue number and view their health records. Real-time data from the crowd-sensing IoT devices will be displayed on the patient's portal so that patients can generate a queue number during less crowded timings.

The doctor portal will be used by doctors. Doctors are able to add data to unique health session records where they add records such as prescribed medication and diagnosis after examining a patient. Researchers portal will allow researchers to view anonymised general health records. Medical Helper portal will be used mainly by counter staff where they can view real-time data from the crowd-sensing IoT device. The IoT crowd-sensing data will also be on the staff portal to help counter staff assign incoming patients to the least crowded waiting rooms, as well as for researchers' use.

There will be access control by user roles for the viewing of patients' records. Doctors can only view patient records after the patient has generated a unique identifier in the form of a QR code for the doctor to view the data. Counter staff are not allowed to view patient records. Researchers can only see a k-anonymity protected version of the records.

1.3 Roles

	Project Management	Web Development	IoT Development	Database Management	DevOps/ DevSecOps	Software Testers
Oscar						
Daryl						
Alicia						
Jie Rui						
Isaac						

Figure 3: Role Table

Oscar Lai - Project Management, Database Management, DevOps/DevSecOps and Software Tester
Oscar is a year 4 information security student with experience database management and devops tools. He will aid in designing and managing the database and to test the software of the application. Lastly, he will also serve as a project manager where he will help plan, organise and direct the completion of issues of the project while ensuring projects deadlines are met and within the scope of the project summary.

Daryl - Web Development, IOT development, Database Management and Software Tester
Daryl is a year 3 information security student who has developed web applications for past projects. He is experienced in database design and web app development. In addition, he will also help out with software testing and IOT development.

Alicia - Web Development, DevOps/DevSecOps and Software Tester
Alicia is a year 4 information security student who has done feature development for a large-scale web app during her recent internship, and small web app side projects in the past. She also has basic knowledge on DevOps tools and processes. Although her security knowledge is not strong, she will also help with software testing.

Jie Rui - Web Development, IOT Development, DevOps/DevSecOps and Software Tester
Jie Rui is a year 3 information security student who has helped to develop web applications for a few projects. Even though he is not as experienced in IOT development and DevOps/DevSecOps, he is willing to learn and help in these areas. Lastly, he will also help out with software testing throughout the course of this project.

Isaac - Web Development, DevOps/DevSecOps and Software Tester
Isaac is a year 3 information security student with experience developing web applications for past projects. Despite his limited experience, he will learn and aid in the use of DevSecOps tools and processes. Finally, he will contribute to software testing as well as hardening of the various systems from his experience in his past internship.

1.4 Timeline

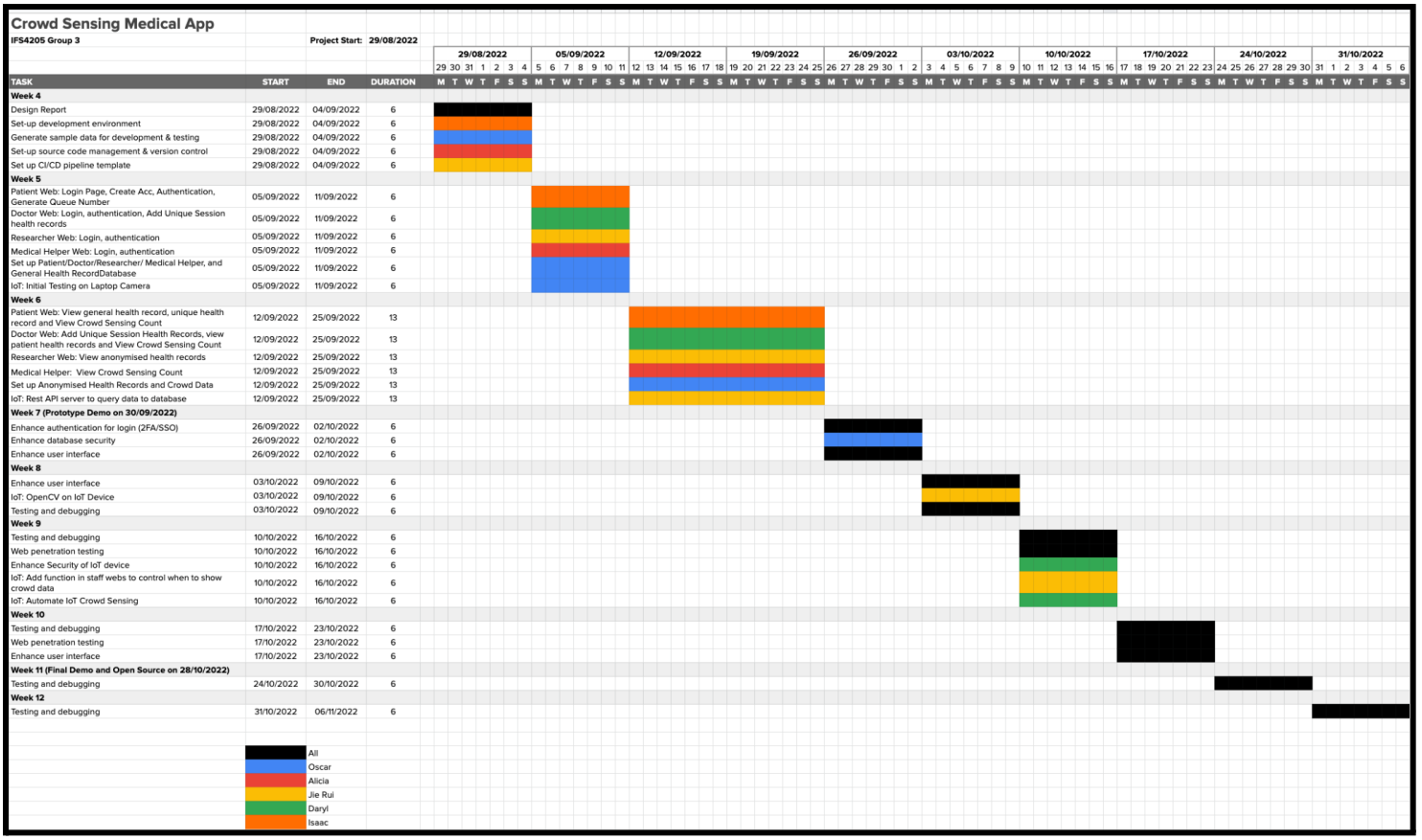

Figure 4: Gantt Chart of Project Timeline

2. Tools

2.1 DevOp Tools

2.1.1 Source Control

	Git	*Perforce Helix Core*
Brief Description	A free, open-source solution, created by the father of Linux, Linus Torvalds. Based on a distributed, decentralised model.	First released in 1995, Perforce Helix was originally called just 'Perforce', and was created by Perforce Software. It's especially well-suited for complex development challenges (such as game development or embedded systems development). And it scales well as teams (and files) grow. Based on a centralised system.
Advantages	● Able to have the entire project history at hand in seconds. ● When a conflict is prompted, there is actually a conflict. Else, git resolves conflicts correctly most of the time and this saves a lot of time. ● A merge is a new commit, which knows what its ancestors are. ● Can be centralised in the future when needed. Able to make changes locally, making local commits, diffs and merges fast. ● Free repo tool.	● Developers can quickly and easily see if they have the latest version of a file on their workstation. ● Any changes saved to the source code are saved in the central server, a central location that manages the versions. Focuses on synchronising, tracking and backing up files. Improves communication as work in progress is easily visible to team members. ● More sensible choice for large repos.
Disadvantages	● If multiple developers are working on the same file when changes are pushed, merge conflicts arise which can be time-consuming. ● Tedious command lines to input and it doesn't track renames. ● Have poor GUI and usability.	● Reported occasions where work was destroyed by P4Merge's auto resolve. ● No way of centralising it in the future. Slower compared to distributed systems. A failure in the central server terminates all versions. ● Every branch is a copy hence if the source tree is huge, disk space gets used up fast. ● Chargeable repo tool.
Final Decision	Git was chosen as many of our team members have had experience with using git in past modules. We feel that we would not have a large number of repos such that we would require the use of Perforce Helix. Most importantly, Perforce Helix is a chargeable tool whereas Git is a free open source tool. We are more inclined to use Git due to its affordability and familiarity. Lastly, to address the issue where Git has poor GUI and usability, we believe that source tree is able to solve that issue as it is a Git GUI that has great usability.	

2.1.2 Build Test Automation

Jenkin
Jenkin is a technology that offers a simple approach to set up an environment of continuous integration as well as continuous delivery. It also supports automation of the development tasks. It offers a rapid approach to integrate the entire sequence to build, test as well as deployment too. Jenkin is also today's leading open source server for automation.

Pros: A user-friendly open source build test automation, easy to configure and modify, available for different operating systems as a platform. It can be configured according to the requirements of CI and CD.

Maven
Maven is a tool for Java development that is utilised for project build automation. It supports mapping out the processes of software building and its dependencies. Maven downloads the plugin and libraries from different resources and then puts all of them on the cache of the local machine.

Pros: It has improved dependency management, hence there is no need to be concerned about transitive dependencies. There is no need to store binary libraries (third party) within the source control and it has better collaboration among source code, plugin libraries and IDE.

Gradle
Gradle is an open source build automation system according to the concepts of Apache Ant and Apache Maven. It determines the order of task execution with the support of a directed acyclic graph. Gradle permits to add and update the build because it fully recognizes the segments of the project that are updated.

Pros: Gradle permits to write the build script with any functional programming language and its tools are user-friendly and easy to use. It is a maintainable and expressive build tool that supports dependency management.

Final Decision: Jenkin
We think that even though Maven is good for dependency management, Jenkin is able to control every stage of the CI/CD pipeline. Having this benefit is instrumental in facilitating a smooth and successful DevOps landscape. Compared to gradle, Jenkin also facilitates with more plugins to support building and testing.

2.1.3 Release Automation

Nexus Repository OSS
Nexus by Sonatype is a repository manager that organises, stores and distributes artifacts needed for development. With Nexus, developers can completely control access to, and deployment of, every artifact in an organisation from a single location, making it easier to distribute software. It supports many formats such as PyPi, Docker, RubyGems and Go.

Pros: Free to use, can be integrated with Jenkins to automate releases and is compatible with popular IDEs such as Visual Studio Code. The free version also includes OSS Index, which provides health check tools that can search the uploaded components against CVEs, analyse the repository for vulnerabilities in open source code used, and produce repository security reports.

JFrog Artifactory
JFrog Artifactory is a universal DevOps solution providing end-to-end automation and management of binaries and artifacts through the application delivery process. It comes with full CLI and REST APIs customizable to the ecosystem.

Pros: JFrog Artifactory provides extensive metadata for both artifacts and folders. JFrog Xray is an open source security scanner which makes use of this metadata to parse linkages between code components, libraries, production applications and projects, among other capabilities.

Final decision: Nexus Repository OSS
Organisations that follow DevSecOps approaches to the SDLC use artifact repositories [vi], hence both Nexus Repository OSS and JFrog Artifactory were considered. The free version of JFrog Artifactory can only be hosted on a public cloud of our choice, which will mean another area to manage when only a temporary place to store code to deploy is required. Although JFrog's provides more open source governance and security scanning capabilities, it may be too sophisticated for a small-scale project. It may also take a long time to learn as it consists of an entire suite of tools such as its own JFrog Pipelines and JFrog Distribution. Since some of our members also have experience using Nexus to deploy code for retrieval from a VM, Nexus Repository OSS was chosen.

2.1.4 Security Testing

The team will be using Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) tools throughout the development of our web application to find security flaws and reduce vulnerabilities.

SAST Tool - SonarQube
SonarQube is a popular Static Application Security Testing (SAST) tool used to scan code for security vulnerabilities. SonarQube's static code analysis supports 17 different languages and provides CI/CD integration. Our team will be using the community edition, which is free and open source.

DAST Tool - OWASP ZAP
OWASP Zap is a popular, free and open source web app scanner. ZAP is a 'man-in-the-middle proxy' and all the information exchanged between the browser and application goes through ZAP. It features active scans, passive scans, fuzzing and more. Its ease in installation and operation will be beneficial to our team members, who have little experience using such tools.

DAST Tool - Indusface WAS
Indusface Web Application Scanner (WAS) is a comprehensive managed risk detection solution which can scan for vulnerabilities and malware. It is a non-intrusive cloud-based solution for scanning and monitoring web applications and systems. Indusface WAS ensures complete coverage of OWASP top 10 vulnerabilities and features remediation guidance to fix vulnerabilities. Indusface WAS provides a free 14 day trial for its comprehensive risk detection features, allowing our team to take advantage of the tool towards the end of the project when our web application is fully deployed.

2.2 Frontend

Languages: **HTML, CSS, JavaScript, jQuery**
Other tools: **React**

Our team briefly considered using a frontend framework such as Angular or Next.js. However, the focus is not on appearance and the only dynamic part of the web apps may be the display of IoT data. Security also depends on implementation. Hence, instead of a full frontend framework, React, a JavaScript library developed by Meta, will be used to create basic user interfaces for the two web apps using the UI components React provides. It does not have a steep learning curve and will help make frontend code more readable and maintainable.

2.3 Backend

The backend facilitates communication between the frontend and our databases. Pandemic security entails the storage of sensitive information in the form of patient particulars and health records. Hence, the foremost priority in the selection of our backend tech stack is security. Our considerations include 3 free and open-source backend frameworks popularly used by developers today.

	Laravel	*Django*
Brief Description	Laravel is a PHP web application framework. It follows the Model View Controller (MVC) architecture and its own Object-Relational Mapping (ORM).	Django is a high-level Python backend web framework. It follows the Model View Template (MVT) architecture and also has its own ORM.
Advantages	● Built-in security features, including CSRF tokens to prevent CSRF attacks and User Input Sanitisation and Validation which protect against cross-site scripting (XSS) ● Easy configuration of its authentication system ● Built-in support for API building ● Integrations with third-party platforms like Amazon Web Services (AWS)	● Good built-in security features, which prevent attacks such as XSS, SQL injections, CSRF attacks and clickjacking ● 'Batteries-included' philosophy means it offers a wide range of features and functionality by importing packages, allowing for reduced coding, greater reusability and faster development ● Easier learning curve due to Python's syntax ● Built-in authentication system ● Better performance than Laravel in a head-to-head test done in 2018
Disadvantages	● Steeper learning curve as compared to Django ● Security features are not as comprehensive as Django	● Have to use library to work around API building ● Routing can be more difficult to learn due to use of regular expressions
Notable Mention	ExpressJS was briefly considered as our backend framework as some of our team members have experience with it. ExpressJS uses JavaScript, which is one of the easier languages to learn, along with Python. However, it falls short in the area of security. Out of the box, it does not offer the comprehensive security features that Django or even Laravel provides. Given the tight timeline, our team prioritised having better in-built security features, rather than having to spend time finding and implementing security packages needed to secure the backend.	
Final Decision	Django was chosen due to its easy learning curve and robust security features which protect from a variety of attacks.	

2.4 Database Server

	MySQL	*PostgreSQL*	*MongoDB*
Brief Description	Relational-based database	Object-based database	NoSQL document-based database.
Similarities	● Supports indexing for performance ● Supports auditing / logs ● Supports TLS/SSL connections ● Works on both Windows and Linux systems ● Role-based access controls ● Point-in-time recovery		
Advantages	● Easy to troubleshoot as it is widely used. ● Good performance for read heavy applications	● Good performance for read-write heavy applications ● Supports advanced data types like IP addresses, arrays, etc. ● Supports more data manipulation operations than MySQL ● Better support for concurrency than MySQL	● Fast as it is document-based ● Built to be scalable
Disadvantages	● Certain features like audit logs and data masking requires enterprise version	● Uses more memory than MySQL	● Some operations (e.g. join) may not be as straightforward as SQL ● May be difficult to use
Final Decision	PostgreSQL was chosen because almost everyone in the team has prior experience using it and it allows for more advanced data types which could come in useful.		

2.5 Crowd Sensing Device/Methodology

Our project requires the use of IoT crowd-sensing devices to be installed in the waiting rooms of the health facility. This device will primarily measure the amount of people that are inside a specific room and relay the message to our staff backend server. This information will then be relayed to our patient and staff frontend page to show the total count of people inside a venue and specific room counts for crowd control measures.

Requirements:

1. Able to measure the number of people inside a venue with >70% accuracy
2. Able to relay the count to the backend server

	OpenCV People Counter	*Pybluez Library*
Brief Description	The people counting camera counts the number of people who arrive, leave or pass through a defined region.	The library allows python code to access the host machine's Bluetooth resources
Advantages	● Available on several programming languages ● Has complexity that meets project scope	● Easy to implement
Disadvantages	● Might not be 100% accurate in counting ● Requires several other libraries/dependencies	● Only available on Python ● Might be difficult to link on an IoT Device ● Might have additional complex layer in ensuring that each bluetooth count is unique
Final Decision	OpenCV People counter was chosen because it presents a good challenge for us as members and the implementation would meet the project scope complexity.	

2.6 Web Server

	Apache	*nginx*
Brief Description	Free, open source web server	Free, open source web server that prioritises speed.
Similarities	● Both web server runs on Linux ● Both web server supports proxy, load-balancing	
Advantages	● Has very comprehensive documentations ● More flexibility in configurations ● Modules can be added to extend functionality	● Consumes less memory than Apache ● Can be used as a reverse proxy ● Faster than Apache
Disadvantages	● Uses more memory than nginx	● Documentations and support may not be as good as Apache
Final Decision	Apache since speed is not a big concern and having more configuration options may come in useful security-wise.	

2.7 Server VM

	Ubuntu Server	*Windows Server*
Brief Description	Ubuntu server is a linux server OS that can be managed via command line interface	Window server provides GUI in managing the server and can be easily accessed through Remote Desktop application
Advantages	● VM server requires less resources ● Easier to use compared to Windows Server ● Set up is easy ● When the Ubuntu operating system needs to be updated, users do not need to restart the machine.	● Provides enterprise functionalities ● Smooth and easy interface
Disadvantages	● Requires CLI experience to efficiently run it	● Requires additional functionalities specific to Windows Server ● GUI server could mean it requires more resources
Final Decision	Ubuntu server was chosen. This is due to our member's familiarity with the system. In addition, the server would require less dependencies and be easier to manage.	

3. Database Design

3.1 Overview

Our database records vital information for our medical facility crowd-controlling application. It mainly comprises 3 parts: user information (data of patients, doctors, medical helpers and researchers), health data (general health records, anonymised patient health records and unique session health records) and our IoT data (Crowd Data) which stores the data collected from our IoT device. In our application, it will store the number of crowds at each specific timestamp. The ER Diagram can be seen in Figure 5.

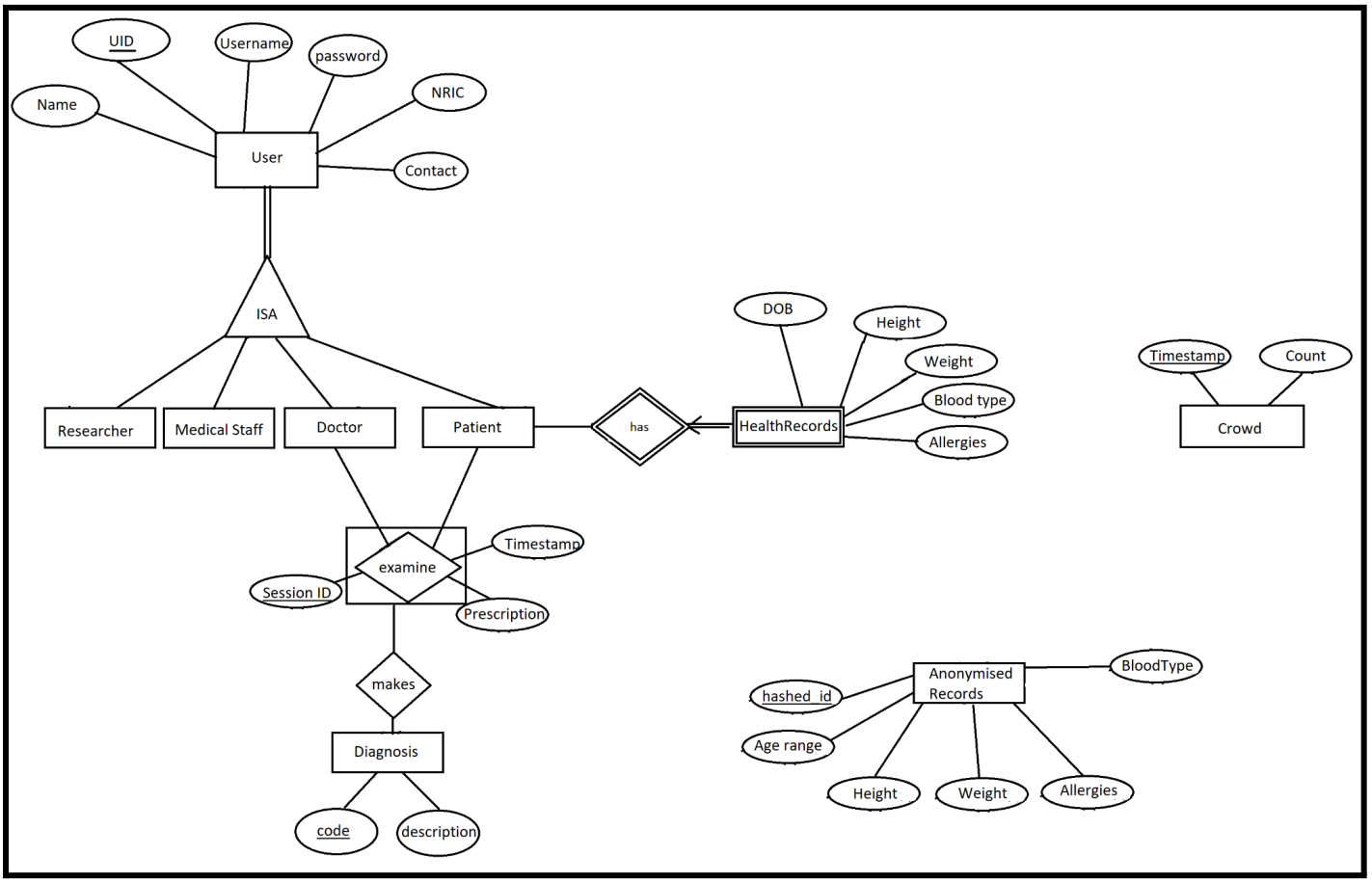

Figure 5: ER Diagram

3.2 Descriptions of tables

3.2.1 User Table

This table is for all users using our web application. This stores the user's login credentials and basic information like their NRIC, name and contact number. The primary key is the unique UID field which is generated every time an account is created.

3.2.2 Patient Table

This table stores the UID of users who are a patient in the facility.

3.2.3 Doctors Table

This table stores the UID of doctors working in the facility.

3.2.4 Medical Staff Table

This table stores the UID of medical staff working in the facility.

3.2.5 Researchers Table

This table stores the UID of researchers who wish to access the anonymized database.

3.2.6 Health Record Table

This table is used to record the general health records of patients such as height, weight, allergies, age and blood type. The primary key is the patient identifier (UID). When a patient creates an account, they are prompted to add in their general health records which is then added as a record to this table.

3.2.7 Anonymised Patient Health Record Table

This table is used to store all anonymised general health records. The primary key is the hashed patient identifier. When a record is added to the General Health Record table, a function in the backend will anonymize the record by converting age to an age range, height to a height range, weight to a weight range and store it to this table.

3.2.8 Patient Visits Table

This table is used to store all patient visits to the facility. The primary key is the session identifier which is generated by the backend, the foreign key is the UID of the doctor, the patient as well as the diagnosis code. When a patient visits a doctor for a medical checkup, the patient will generate a session identifier which is used by the doctor to insert a new record of the session details such as timestamp, prescription given and diagnosis code to the table.

3.2.9 Diagnosis Table

The table is to store the diagnosis codes that doctors will use when attending to a patient. Each diagnosis code will also have its description.

3.2.10 Crowd Data Table

The table is to store all crowd data measured by the crowd-sensing IoT device. The primary key is timestamp. When the crowd-sensing IoT is live, the crowd-sensing IoT device will measure the amount of people in the medical facility every 10 minutes interval and insert the record to the table.

4. References

Section 1.1
https://doi.org/10.3389/fmicb.2020.631736

Section 2.1.1
https://www.educba.com/perforce-vs-git/
https://www.educba.com/introduction-to-git/
https://www.clearvision-cm.com/blog/git-vs-perforce-helix-whats-the-difference/#:~:text=One%20of%20the%20key%20differences,other%20hand%2C%20can%20be%20centralised.
https://www.perforce.com/blog/vcs/git-vs-perforce-how-choose-and-when-use-both

Section 2.1.2
https://www.javatpoint.com/gradle-vs-jenkins
https://solutiondots.com/blog/technology-blog/jenkins-vs-maven-vs-gradle/
https://www.browserstack.com/guide/maven-vs-jenkins

Section 2.1.3
https://www.plutora.com/ci-cd-tools/artifacts-management-tools/nexus
https://www.sonatype.com/products/repository-oss-download
https://www.sonatype.com/products/nexus-repository
https://www.sonatype.com/products/repository-health-check-ossi
https://www.jfrog.com/confluence/display/JFROG/JFrog+Artifactory
https://www.techtarget.com/searchsoftwarequality/tip/Sonatype-Nexus-vs-JFrog-Pick-an-open-source-security-scanner

Section 2.1.4
https://blog.gitguardian.com/security-tools-shift-left/
https://www.softwaretestinghelp.com/dynamic-application-security-testing-dast-software/#1_Indusface_WAS_Recommended_Tool
https://bandit.readthedocs.io/en/latest/
https://www.sonarqube.org/features/security/
https://www.zaproxy.org/

Section 2.3
https://statisticsanddata.org/data/most-popular-backend-frameworks-2012-2022/
https://blog.back4app.com/backend-frameworks/
https://www.raftlabs.co/development/most-popular-backend-framework-of-2021
https://www.cloudways.com/blog/laravel-security/
https://technicalistechnical.com/laravel-framework/
https://www.benchmarkit.solutions/lets-understand-the-pros-and-cons-of-using-django/
https://data-flair.training/blogs/django-advantages-and-disadvantages/
https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/web_application_security

Section 2.5
https://pyimagesearch.com/2018/08/13/opencv-people-counter/