



THE UNIVERSITY OF
SYDNEY

School of Computer Science UG Capstone Projects

COMP3888

SetDose - Optical recognition software for liquid medication dosing

Group: COMP3888_T15_06_Group4

Client name: Dr. Rebekah Moles, Philip Boughton

Tutor: Regina Wang

Group members details

- Xiaolin Zhao, 480406662, xzha3077
- Zhiran Bai, 480446772, zbai6612
- An Yan, 480158532, ayan0137
- Zhiyang Zhang, 470482647, zzha6434
- Hurui Yang, 480170936, hyan5496

14th November 2021

Table of Contents

<u>ExecutiveSummary</u>	3
<u>Introduction</u>	4
<u>System Specifications and Design</u>	9
<u>Quality of Work</u>	31
<u>Quality of Group Process</u>	64
<u>Reference</u>	81
<u>Code base</u>	82

Executive summary

Our project title is SetDose - Optical recognition software for liquid medication dosing. And we have five members in our group, which are An Yan, Zhiran Bai (Robert), Xiaolin Zhao, Hurui Yang (Warren), Zhiyang (Alex). Here is the introduction for each of us:

- An Yan is currently studying a major in Computer Science with a Bachelor of computing. He can use Python, Java, and C to develop some basic programs and solve some problems. And An has learned some basic algorithms and computer vision knowledge through previous units. Also, An is currently learning machine learning this semester and has enough passion to do machine learning. Therefore, An is confident in completing this project and will devote himself to it.
- Robert's major is Computer Science, which determines what problems can be solved with computers and uses complex algorithms to solve them. Robert's computer science background has provided him with the ability to undertake quantitative analysis using large-scale data sets and software coding techniques, as well as soft skills like perseverance and learning skills. These skills will be highly valuable during the development of this project.
- Zhiyang (Alex) is currently studying a major in computer science and a minor in software development. Alex is proficient in Java, Python, and C programming language and is familiar with algorithms and machine learning. Alex's courses provide him with different beneficial skills including image processing, UI designing and programming, and application development. Though machine learning is an area where Alex is not familiar with, Alex believes his capability of learning could satisfy the needs of learning this technique.
- XiaolinZhao is studying a major in computer science and a minor in mathematics. He can proficiently use the knowledge of C, Python, and Java.
- Warren is currently studying a major in computer science and a minor in data science. Warren is good at Java, python, c, c++, and go, also is familiar with algorithms and machine learning. Warren's courses cover basic algorithms and some knowledge about AI and also write some basic programs. Warren believes that good cooperation is based on better communication.

Our system is designed to meet the needs of parents, medical professions, and other groups to measure the medicine in the syringes. Specifically, we will develop an android app that can identify the type and volume of medicines, so as to help our target population feed drugs to children more safely and scientifically.

For our project, we would need some kind of discipline knowledge. Obviously, because we develop our program in Python, we need knowledge about Python. And of course, we need to know computer vision to deal with images. And we choose to do machine learning to recognize color in syringes so we need to have knowledge about machine learning like Pytorch and Tensorflow. At last, we need to know the python framework for Web and Android to implement these functions on web and Android devices.

1. Introduction

1.1. The overall project vision, goal, purpose, or objective

Pharmacists adjust dosing and perform medicine injections for patients every day. Yet according to *Eyler (2021) [1]*, measuring the correct dosage is one of the most challenging parts of the prescribing process. The reason for this is due to the complexity of an individual's health condition, body function, and medicine itself. Sometimes, medicine itself can be hard to manage due to its narrow therapeutic index and wide interpatient variability (*Sharabiani e.t, 2018) [2]*. Thus, one must consider the patient's condition, age, organ function, comorbidities, and concurrent medications in order to give out the correct dosage.

But the problem does not only happen in the chemist's shop, it also happens commonly in hospitals. According to the *Australian Commission of Safety and Quality in Healthcare (2020) [3]*, 20% of medication-related adverse events occur in hospitals. More than one-fifth of individuals were readmitted to hospitals due to medication-related issues.

The same issue arises in households as well as in the hospital. According to *the Children's Hospital of the king's daughters(2017) [4]*, before any subcutaneous injections at home, the parent/family member should always check the label, medication type, concentration, expiration date, and medicine color to ensure medicine quality. Still, it has been found that 50 million out of 3 billion sample families in the *USA have encountered medicine and dosage problems (Aronson, 2009) [5]*.

Thus, there is a clear need for hospitals and pharmacies to implement a verification tool that is accurate, easy to use, and accessible to mitigate these issues.

So, the purpose of this project is to focus on the medication dosing area with a particular focus on dosing and medication recognition to provide the medical industry and syringe-related personnel with a tool that could assist the dose-measuring and prescribing process.

The goal of this project is to use our individual knowledge and disciplinary strengths to build an optical recognition algorithm and then use the algorithm to implement a website that has the function of recognizing syringe images and being able to store and manage recognition history and an administering account that can manage all the users. After that, an android application will be developed with a minimalist user interface and simplified functions of syringe recognition to meet the needs of the user group that is computer-challenged.

The objectives are here blow:

1. examine and research on current application, solution, and programming language libraries about image recognition and syringe recognition algorithm
2. Use the knowledge gained from research to build the colour recognition algorithm and volume recognition algorithm
3. Improve the accuracy of colour recognition algorithm and volume recognition algorithm
4. Integrate the developed algorithm into one function for easier development of website and android application
5. Build a database for recording recognition history, user information, and administering accounts.
6. Implement the front-end of the website using HTML and CSS.
7. Implement flask API to connect the front-end and back-end of the website.
8. Implement the select function in the android app
9. Implement the UI for the android app
10. Integrate the recognition algorithm in the android app

The vision of our project is to create an online tool and an offline tool of syringe usage tracking for both medical professionals and home syringe users. For medical professionals, our website will let them have a better knowledge about their patient's health conditions, essential information, and past--current medicine usage. With the help of these pieces of information, the professionals will be able to calculate the optimal ratio for the medicine to better fit the patient's body and through the history feature, the professionals will be able to track the actual use of the medicine, thus medicine will be able to adjust according to the usage, in this way, the patient will have less chance to encounter side effects and recover at the optimal rate. For home syringe users, Our website and application will record and verify the syringe colour and volume every time, so these tools will act as an assistant tool and a second confirmation to help the users to minimize human error and keep better track of their usage.

1.2. What the project will achieve

This project will achieve creating a syringe recognition algorithm, a website that can be able to recognize, store, manage syringe information, and an android application that has a simplified user interface for computer-challenged users to use.

For the syringe recognition algorithm, we will implement the algorithm using 2 neural network packages “tensor flow” and “pytorch” to achieve the medicine recognition part of the algorithm. This medicine recognition part will be able to identify the medicine type from the syringe image that the user uploaded. The other part of the algorithm is using the “OpenCV” packages to achieve the function of syringe volume recognition. We will use the OpenCV package to first locate the whole syringe in the image and then we will size the syringe in a rectangle shape to know the ratio between the real syringe and the one in the image then we will also locate the medicine in the image using the same method. After that, we will be able to get the exact medicine type and medicine volume.

For the website, we will implement the recognition algorithm and we will take an account-based interaction strategy where each user will need to create an account in order to use the website. On the website, users need to upload their syringe image on the home page, and then, after clicking the upload button, the website will show the result. Users can find all their recognition history in the bottom section of the profile page. In the history section, the website will record all the essential information about the recognition, including time, date, medicine, volume. In the upper section, the user information will also be displayed, this user information is taken from the account registration process and if the user wants to change any information, we also have a web page for profile editing.

For the android application, we are focusing on creating an application that is offline and easier to use. The application has the function of selecting photos from the phone and then the app will return the medicine type and volume. This application is designed to meet the need of the computer-challenged user group and for environments that don't have internet.

1.3. The key stakeholders, what do they do, and how they interact

The Key Stakeholders of this project include nurses, pharmacists, patients with at-home injection needs, hospital executives, Sydney Pharmacy School - SetDose Team.

For nurses and pharmacists, they are the end-users of the website and software. They will use our tools under real application scenarios and gain direct benefits from the software. From the interaction with the website and android app, it will let them have a better knowledge about their patient's health conditions, essential information, and past--current medicine usage. With the help of these pieces of information, these professionals will be able to calculate the optimal ratio for the medicine to better fit the patient's body and through the history feature, the professionals will be able to track the actual use of the medicine, thus medicine will be able to adjust according to the usage, in this way, the patient will have less chance to encounter side effects and recover at the optimal rate.

For patients with at-home injection needs, Our website and application will record and verify the syringe colour and volume every time, so these tools will act as an assistant tool and a second confirmation to help the users to minimize human error, keep better track of their usage and provide useful feedback and medicine trajectory for the medical professionals.

For hospital executives, they can gain indirect benefits from introducing the website and android application to the hospital. By using our website and android app, it will help the medical worker in the hospital to reduce the in-hospital dosing error rate and prevent hospitals from potential lawsuits due to dosing mistakes.

For the SetDose team, this project will provide them with potential approaches/solutions for the implementation of the website and android app, which will in a way help them avoid mistakes and gain useful information about the developing process of this project.

1.4. Identification of resources and risks involved in the project

The resources that we consider to be beneficial for this project includes support from supervisors, online documents(i.e. TensorFlow, PyTorch, machine learning, existing neural network structure), youtube videos, online discussion forum (i.e. stack overflow forum), group member's pre-existing knowledge.

For the supervisor, this includes the tutor and client. From our tutor, we gained useful feedback for our presentation, report, and weekly discussion. The tutor not only acts as a marker for our assignments but also acts as a tracker and adviser that reminds us of the progress we should be making and gives us advice about our group work. From our clients, we got useful information and advice about them. Since the clients are familiar with the context of this project, they have given us technical support and recommendations for the method and tool that can be used for the project.

From online documents, we can find detailed information about one specific programming library. By viewing the document, we can learn the syntax, functions, features of the library and build what we intended to build. I think the online document is one of the most powerful tools we used during the development of the process and I believe the most important skill that a programmer should have is being able to read and understand online documents.

For youtube videos, the video works similar to online documents but it conveys more information and code examples for us to learn from. The video usually explains things that are in the online document and will provide demonstrations that are related to the particular function mentioned earlier.

From online discussion forums, we use this tool more on finding solutions to error codes. For error codes, we can learn from other people's mistakes. The error we met definitely appears before in other people's code, thus we can learn how they fix the problem and apply it to our situation.

For group members' knowledge, we use collective knowledge to decide our project's direction and fix code errors. This is the most valuable resource we have.

The risk we identified includes external risk (client's requirement changes), system internal risk (security), technical difficulties, changing priorities, lack of communication, unplanned absence of a team member, failure to deliver on time.

For external risk, we mainly considered the client's requirement changes. For example, in the latter part of this semester, the client wants us to develop an offline mobile application for an environment that doesn't have internet and signal. In this scenario, we have to reschedule our working plan and make time for developing the mobile application.

For system internal risk, we mainly considered web and database security. So during the development of our website, we considered the scenarios of flooding attacks and we set a mechanism to handle the risk.

For technical difficulties, the protocol is: we first try to read the online documents for a solution, if we don't have an answer, we will use all the other resources mentioned above to try to find an answer.

For changing priorities, lack of communication, absence of a member, and failure to deliver, this is all related to communication strategy. The way we used to prevent these risks is to have meetings once every 2 days. So in this case, we will be able to find solutions before things are too late.

2. System Specifications and Design

2.1. User stories

2.1.1. User story1: As a general-public user, I want the software to check the syringe so that I can make sure I used the right dose

2.1.1.1. Acceptance Criteria

Given the image of the syringe with liquid medicine in it, upload the photo of the filled syringe, ensure the right syringe size of the dose, and the right medicine volume is displayed.

2.1.1.2. Functional requirements

The accuracy for the volume detection should be satisfyingly high enough, e.g. above 75%.

2.1.1.3. What we achieved

The volume can be detected with approximately 75% accuracy.

2.1.1.4. Testable

Remain a part of the data set using as test data to test the validation of the algorithm, swap the test data, and train data to do more tests and overall accuracy is above 75%.

2.1.2. User story 2: As a general-public user, I want the software to check the medicine inside the syringe so that I can make sure I used the right medicine

2.1.2.1. Acceptance Criteria

Given the image of the syringe with medicine in it, when uploading the photo of the filled syringe, then ensure the right medicine colour and medicine name is displayed.

2.1.2.2. Functional requirements

The accuracy of the medicine detection should be satisfyingly high enough, e.g. above 75%.

2.1.2.3. What we achieved

The medicine name can be correctly determined based on the colour with approximately 75% accuracy

2.1.2.4. Testable

Remain a part of the data set using as test data to test the validation of the algorithm, swap the test data, and train data to do more tests and overall accuracy is above 75%.

2.1.3. User story 3: As a general-public user, I want the program to recognize images as soon as possible

2.1.3.1. Acceptance Criteria which we satisfied

The software will return the result within 2 seconds.

2.1.3.2. Testable

Test different functions with different data, all the results should be returned within 2 seconds.

2.1.4. User story 4: As a general-public user, I would like to have an account to log in to the web application, for better security and more functions such as the history function.

2.1.4.1. Acceptance criteria

Users should be able to register an account and manage their account details.

2.1.4.2. Functional requirements

A database is needed to store all the account details.

The registration interface should be easily usable.

2.1.4.3. What we achieved

Users could easily register an account and log in to our system with it.

2.1.4.4. Testable

Ask real users to complete conducted tasks using our system, the test would pass if we observe that they could easily complete all the tasks without any guidance.

2.1.5. User story 5: As a general-public user, I want to review the past record so that I can keep track and manage my dose

2.1.5.1. Acceptance Criteria which we achieved

- * Be able to view history records in the software.
- * Be able to remove history records.
- * Be able to Add date to each record when measuring.
- * Be able to navigate to a record detail page (which may include drug colour, name, type, dose, date)

2.1.5.2. Functional requirement

Have a database storing the upload history including the medicine, volume, comments, and images.

2.1.5.3. Testable

Conduct test plans to test the history function works as expected.

2.1.6. User story 6: As a general-public user, I hope I can add and change personal information for privacy security and get advice from authorities

2.1.6.1. Acceptance Criteria which we achieved

Web applications should have features that allow users to easily change their personal information and make it inaccessible to anyone except administrators.

2.1.6.2. Functional requirements

Connect to a database storing users' personal details.
Have administrator accounts to manage all the normal users.

2.1.6.3. Testable

Conduct the test plans to test the functionality between website and database works as expected.
Also, test the administrator could access and make changes to the normal users' accounts details

2.1.7. User story 7: As a general-public user, I want to use the recognition function more easily and conveniently anywhere

2.1.7.1. Acceptance Criteria which we achieved

The lighter version of Android will allow users to use the recognition function on mobile phones in the offline environment.

2.1.7.2. Functional requirement

Develop an Android application that can be used offline to achieve the same functionality as our website.

2.1.7.3. Testable

Conduct test plans to test the Android application could run smoothly and generate accurate results.

2.1.8. User story 8: As a pharmacist, I want to use the software offline so that I can easily perform the measuring from anywhere

2.1.8.1. Acceptance criteria which we achieved

When working in an offline environment, the lighter version of the Android app should display the correct volume, colour, medicine name

2.1.8.2. Functional requirements

The offline android app could achieve the same functionality as our online website.

2.1.8.3. Testable

Use a portion of our test set to test the accuracy of the result from the offline android application.

2.1.9. User story 9: As a general-public user, I want the software to be easy to use so that I can use it correctly

2.1.9.1. Acceptance Criteria

Display the dose result correctly through algorithms Display the color result correctly through algorithms *Display the medicine name

2.1.9.2. What we achieved

All the results could be correctly displayed

2.1.9.3. Non-functional requirements

The results should be easy to access and easily readable.

May consider the access for disables

2.1.9.4. Testable

Conduct a user survey for a group of people from our targeted users to test the usability of our result display.

2.1.10. User story 10: As an administrator, I want to manage all users and have the right to handle abnormal users.

2.1.10.1. Acceptance criteria which we achieved

Web applications should have an administrative center that only administrators with specific permissions can access, allowing administrators to change user profiles and delete users

2.1.10.2. Functional requirements

Implement the functionality of administration including editing normal users' account details, editing normal users' upload history, and deleting normal user accounts.

2.1.10.3. Testable

Include testing the administration functionality into the test plans of the website.

2.1.11. User story 11: As a data scientist, I want to use the website via the web browser.

2.1.11.1. Acceptance criteria which we achieved

Correct results should be generated and displayed correctly through the website.

2.1.11.2. Functional requirements

The website could work without causing errors and correctly output the result generated by the algorithms.

2.1.11.3. Testable

Use website testing technology (Selenium) to test the overall performance of the website and check its result accuracy.

2.1.12. User story 12: As a data scientist, I want to use the offered APIs to recognize the images.

2.1.12.1. Acceptance criteria which we achieved

There will be some documents of APIs including request example and response example.

2.1.12.2. Non-functional requirements

API documentation should be formally readable and understandable by professionals.

2.1.13. User story 13: As a pharmacist, I want to obtain the user's personal contact information with the permission of both the user and the administrator to provide professional medication advice

2.1.13.1. Acceptance criteria which we achieved

Web applications allow administrators to provide limited contact information to pharmacists with the user's consent.

2.1.13.2. Functional requirements

Administrator accounts could have access to the normal users' personal details for contact.

2.1.13.3. Testable

Include checking normal users' account details into the website administration test plans.

2.1.14. User story 14: As a general-public user, I want the server to recognize images as soon as possible and return results fastly

2.1.14.1. Acceptance criteria which we achieved

The server will return a result within 2 seconds.

2.1.14.2. Functional requirements

The server will return a result within 2 seconds

2.1.14.3. Testable

Manually test that the result could be displayed within 2 seconds using the website.

2.2. System design

2.2.1. Overall System design of openCV

2.2.1.1. General concept of formula

Generally speaking, the main feature of the SetDose project is to recognize the type of medicine and the volume in the syringe. Based on these two features, we do some technical research. The final conclusion is that the computer version is the traditional way of recognition. The computer version(CV) mainly uses openCV. The openCV is written by C++ and offers APIs in Python and Java. The CV solution can do two parts:

- a. Get the total volume of the syringe and the position of the piston of the syringe.
- b. Get the background colour and the colour of liquid medicine in the syringe.

For this part we can use simple division to get the volume by the formula:

$$v = \delta \times \omega$$

δ is the proportion of non-liquid and liquid. For the general case, δ is a fraction.

ω is the total volume of the syringe. The unit of the syringe is not certain, both litre and millilitre are acceptable. The reason for this is the unit of the syringe is on the side of the syringe. If we take images from the front of the syringe, then the unit will be much vaguer. Thus, we only return the value without a unit.

2.2.1.2. The details of getting the parameter in the formula.

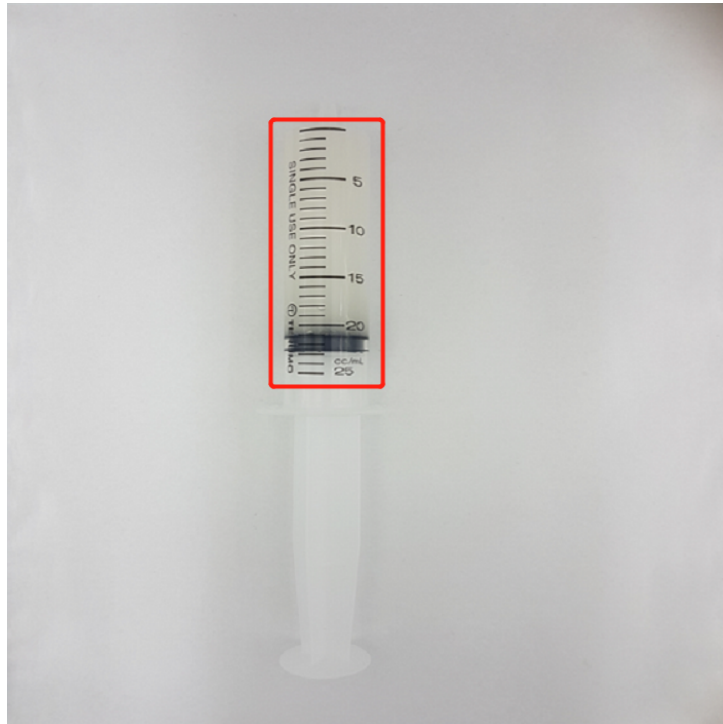


image1: the screenshot of the recognition

Image 1 illustrates the way to get δ and ω :

1. read the image from the local disk
2. Convolve the source image with the specified Gaussian kernel using function `GaussianBlur()`
3. Find edges in the input image and mark them in the output map edges using the Canny algorithm.
4. Convert to the whole binary image. The definition of the binary image is that the colour of the image only contains two colours: white and black.
5. Based on the binary image in step1, create a rectangle containing all pixels of the syringe. The rectangle could be large or small.
6. Create a line on the piston. The position of the line is exactly on the top of the syringe because the needle of the syringe is always on the top of the image.
7. The line in step3 cuts the whole rectangle into two parts. By using simple math knowledge, we can know the area of the two rectangles and the total area of the large rectangle.
8. For the two rectangles, the openCV will recognize the part that the colour of that part does not fit the background colour. That rectangle will be marked as the rectangle containing medicine.

9. Then, by simple calculation, we can get the percentage of area with the medicine of the total rectangle. So, we get δ .

The way to recognize the total volume of the syringe is similar to the way of recognizing the volume. After converting the whole image to a binary image, the numbers on the syringe can be recognized directly.

For now, we get the total volume and the proportion of non-liquid and liquid. We can calculate the actual volume of the liquid medicine based on the two parameters.

2.2.1.3. The way of recognizing the type of medicine

We have three types of medicine: Advil, Demazin and Panadol. The corresponding colour for the three medicines is transparent, black and red. After we have determined the colours of the three medicines, we can realize the medicine identification by identifying the colour of the liquid in the syringe.

The key step of recognition is:

1. Convert the image from RGB image to HSV image.
2. Go through all pixels in the image to identify the colour of the liquid in the syringe. The way of identifying it is to test the RGB value of the pixel whether in the range of blue or the range of red.
3. If the colour of medicine is not in the range of blue and red, then we can conclude the colour of medicine is white or close to white.

For now, we get the type of medicine and the volume of liquid in the syringe.

2.2.2. Overall system design for machine learning

(The aim of both TensorFlow and Pytorch is to recognize the type of medicine.)

2.2.2.1. The main recognition step of Pytorch(using LeNet5 model)

1. We first preprocess the data, including re-changing the size of the picture, data augmentation and standardizing the pixels in the picture, and so on. These data pretreatment technology can well adapt to let the data and model, and the standardization of the pixels in the image and enhance data can increase the richness of data sets, let model come into contact with more abundant data,

which can make a model deal with more complex scenarios, and avoid overfitting phenomenon of the model.

2. Secondly, we divide the pre-processed data into the training sets and test sets. The training set is used for the training model, and the test machine is used for evaluating model
3. Thirdly, we set up the LeNet5 model, which is one of the oldest convolutional neural network machine learning models. Although it is very simple, it has high accuracy and high training efficiency for the situation where we need to classify only three colors. In order to make it more convenient and integrated, several convolutional layers and pooling layers form a forward function and superimpose several such functions in the model to obtain the final model
4. Fourth, we use the training set that has been segmented above to train the model. Among them, we can adjust the learning rate and epoch in model training, because different learning rates, batch sizes, and epochs will affect the training efficiency and accuracy of model training. Generally, the higher the learning rate, the faster the convergence speed, but at the same time, it is easy to produce an over-fitting phenomenon. However, if the learning rate is too small, the over-fitting phenomenon will be avoided, but the convergence speed will be slow, and the optimal effect may not be achieved in the end. The size of batch size will affect the accuracy of training, but generally speaking, the smaller the batch size will make the model more accurate. Finally, the amount of epoch also determines the performance of the model. Too little epoch will make the model stop training before it is trained to the best performance, while too much epoch will prolong the training process and the model is prone to overfitting.
5. Finally, The accuracy of these models with different parameters was tested on the test set to evaluate the expressiveness of different models, and the model with the highest accuracy was selected as the best model, which was saved to predict the color and type of drugs in the syringe.

2.2.2.2. The main recognition step of Tensorflow

1. We first preprocess the data, including re-changing the size of the picture, data augmentation and standardizing the pixels in the picture, and so on. These data pretreatment technology can well adapt to let the data and model, and the standardization of the pixels in the image and enhance data can increase the richness of data sets, let model come into contact with more abundant data, which can make a model deal with more complex scenarios, and avoid overfitting phenomenon of the model.
2. Secondly, we label the pre-processed image and put these images into a data bundle using “pickle”.
3. Thirdly, we set up the convolutional neural network model. It includes several convolutional layers, pooling layers, flattened layers, and dense layers to form a model. And used the tensorboard to find the best network configuration.
4. Fourth, we used the “model.fit” function to run the image bundle with our neural network.
5. Finally, The accuracy of these models with different parameters was tested with tensorboard to evaluate the accuracy of different models, and the model with the highest accuracy was selected as the best model, which was saved to predict the colour and type of drugs in the syringe.

2.2.3. Web System design during the whole process

2.2.3.1. System design at the beginning of the project

(From the beginning of the project, we divide the whole project into four parts)

2.2.3.1.1. The core algorithm and cross-validation

The core algorithm contains the computer version part and machine learning part. The computer version part will use OpenCV and the machine learning part will use Tensorflow and Pytorch. After that, all results will be done in cross-validation. The strategy of cross-validation is:

- a. if all three algorithms get the same result, then return that result
- b. if both two algorithms get the same result, then return that result.

- c. if all three algorithms get different results, then redo the recognition part and re-enter the cross-validation until getting the result.

2.2.3.1.2. The delivery part of the project

For the delivery part, we want the server and Android app to be in parallel. They offer the same recognition service, while the server offers online service and the Android app offers offline service.

a. The system design of the website

The main purpose of the website is to offer services for consumers. The website should be clear and easy to use.

For agile development, the website does not separate the front-end and back-end. The back-end is written in python. The framework we use is Flask. The reason for that is Flask gives the developer varieties of choice when developing web applications, it provides tools, libraries, and mechanics that allow you to build a web application but it will not enforce any dependencies.

Flask offers some templates. These templates include basic HTML and variable delivery. There is no CSS and js code in the template. We use external CSS and js resources in “cdn.jsdelivrivr.net”. Also, we do not separate the CSS and the js code because the CSS code is not large enough to separate.

b. The key feature of the website

We have mentioned that the key function of the website is the recognition of liquid medicine in syringes. In advance, we add two main features:

- 1. Permission control.*** Because we want to control the upload images and ensure the security of the website, we add permission control via add register control. Everyone should register an account and use that account to visit our website. All registered accounts are normal accounts. Meanwhile, we have a root user to control all other accounts. This root account can set an account to be in use or not in use. If we find some strange behavior of the account such as uploading too many images or uploading illegal images, we will set the state of the account to be

not in use. Then, the owner of the account should contact us for further help.

The register details are some basic information including name, email, gender.

- II. *History images review.*** We have required all users to register an account before uploading and recognizing. For the account, we can attach the upload images to the account, so that users can review their upload history. We limit the history to be 7 days due to the storage limit of the database. Also, because the size of some images is too large, we also store the images in Redis as cache. The read speed of Redis is much faster than the traditional database.

c. *The key feature of the Android app*

We transplanted the core algorithm to the mobile phone. To cover more groups of people, including the old group and the group with low knowledge of the smartphone, the main concept of the app is the simpler, the better. We prefer the app to be used offline. Because Android does not have a mechanism like iOS that if users want to install an app, it should be downloaded via AppStore, thus, we can distribute the Android apk package and abb package which is the updated format of installing the package. The install package could be downloaded and installed without any verification of staff.

We also prefer the app to be free for all users. We have announced that the app will use the GNU General Public License. The core concept of the GNU GPL license is that everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. Also, the app can not be in commercial use. It is completely free.

The app is written with API 26, which can cover the version of the Android app from 4.0 to 12. The app is backward compatible and if some smartphones can not run the app due to the change of the operating system of smartphones, they can also visit our website and upload images inline.

The main design of the app is that no design is the best design. We have added some text to tell the users how to

use the app, but for someone who does not know it well, we have set that only one button on the whole screen could be clicked. After clicking the button, the users will select the images from the local photo album and the result will appear on the screen. The whole process is easy to use and comprehensive.

2.2.3.2. System design change during the development

2.2.3.2.1. The database selection

At the beginning of the project, we plan to use MySQL as our database. MySQL is a relational database management system (RDBMS), which means it stores records in multiple, separate, and highly codified tables rather than a single repository. There is 5 reasons for using MYSQL:

1. Open-source and compatible. This simply means that anyone can install and use the basic software, while also enabling third parties to modify and customize the source code. More advanced versions, which offer additional capacity, tools, and services, come with tiered pricing plans. MySQL is also built to be highly compatible with a wide range of systems, programming languages, and technologies, including alternative DBMS solutions.
2. Fast and reliable: MySQL was developed for speed, and maintains a reputation for being fast, even if this may come at the expense of some additional features. Another benefit is that it is relatively simple to learn and use.
3. Availability. Online businesses and web platforms need to be able to provide round-the-clock services for a global audience, and high availability is a core feature of MySQL. It uses a range of cluster servers and data replication configurations that ensure uninterrupted uptime even if there is a failure.
4. Scalability. As data volumes and user loads increase, the database needs to be scaled up to cope with the additional workload without a drop in performance. MySQL can be scaled in different ways, typically via replication, clustering or sharding.
5. Security. MySQL offers encryption using the Secure Sockets Layer (SSL) protocol, data masking, authentication plugins, and other layers of security to protect data integrity.

For the above 5 reasons, we decided to use MySQL as the database. However, after we talk with our clients, they suppose we should use a database as lite as possible. We do not want to abort the 5 key features of the database, so we select a liter database. There are 2 popular lite databases: H2 and SQLite.

- H2 is a relational database management system written in Java. It can be embedded in Java applications or run in client-server mode. It is open-source and A subset of the SQL standard is supported.
- SQLite is a relational database management system (RDBMS) contained in a C library. In contrast to many other database management systems, SQLite is not a client-server database engine. Rather, it is embedded into the end program.

Both SQLite and H2 support SQL language and the SQLite engine has no standalone processes with which the application program communicates. Instead, the SQLite library is linked in and thus becomes an integral part of the application program. For this reason, even though the system coupling could increase, we still decided to use SQLite.

The SQLite will store the history images and the user information including username, email, gender, and some basic information. The id of history images will be recorded in SQLite. The actual image will be stored in the Redis.

2.2.3.2.2. no-SQL database selection

Because the size of images is quite large, after discussing with our clients, we plan to store them in a no-SQL database such as redis. Redis is an in-memory data structure store, used as a distributed, in-memory key-value database, cache, and message broker, with optional durability. The reason to select Redis rather than MongoDB is:

1. Redis has high performance and linear scalability of up to 1000 nodes. There are no proxies, asynchronous replication is used, and no merge operations are performed on values.
2. Redis has an acceptable degree of write safety: the system tries (in a best-effort way) to retain all the writes originating from clients connected with the majority of the master nodes. Usually,

there are small windows where acknowledged writes can be lost. Windows to lose acknowledged writes are larger when clients are in a minority partition.

3. Redis has high Availability: Redis Cluster is able to survive partitions where the majority of the master nodes are reachable and there is at least one reachable replica for every master node that is no longer reachable. Moreover using replicas migration, masters no longer replicated by any replica will receive one from a master which is covered by multiple replicas.
4. Redis is easy to deploy and more popular. The GUI of Redis is user-friendly and easy to use.

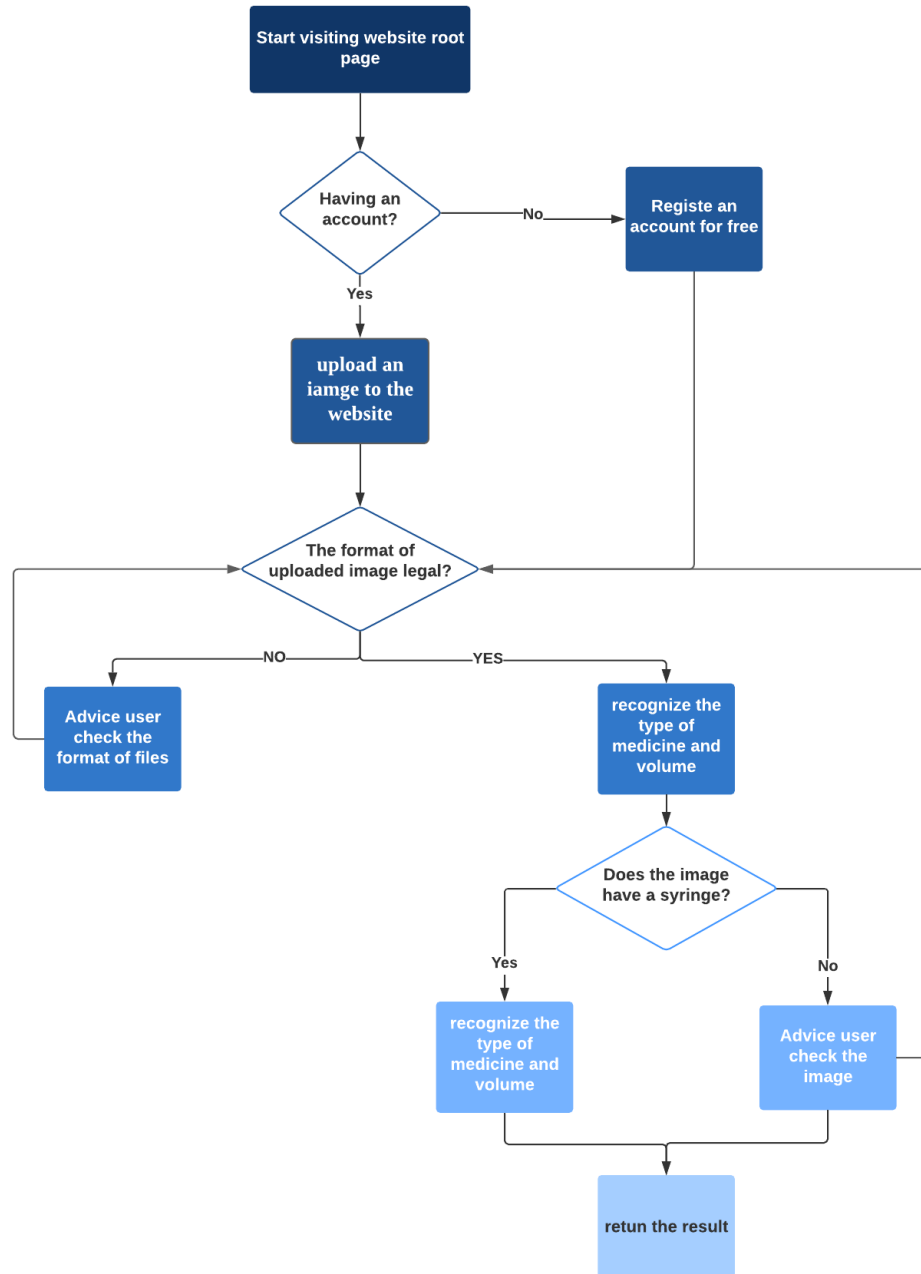
2.2.3.2.3. Android app framework

At the beginning of the project, we plan to use Beeware as the framework for the development of the Android app. The Beeware is written in python. The reason for using Beeware is:

- Python framework VS Java Kotlin
Our core algorithm is written by python, so if we also develop an Android app in python, we can apply our core algorithm directly and no more other code. The advantage is no extra work to be done by using BeeWare. However, there are some disadvantages. BeeWare is still in the early stages of development and doesn't have full support for Android yet. For instance, if we want to add a button in Beeware and write some style code. The button is not supported in Android 4.0. We have mentioned that we want the app to support a more Android version, so we departed Beeware and decided to rewrite the core algorithm in Java using OpenCV. OpenCV also offers APIs in Java and kotlin.

2.3. System architecture

2.3.1. Workflow of web request processing

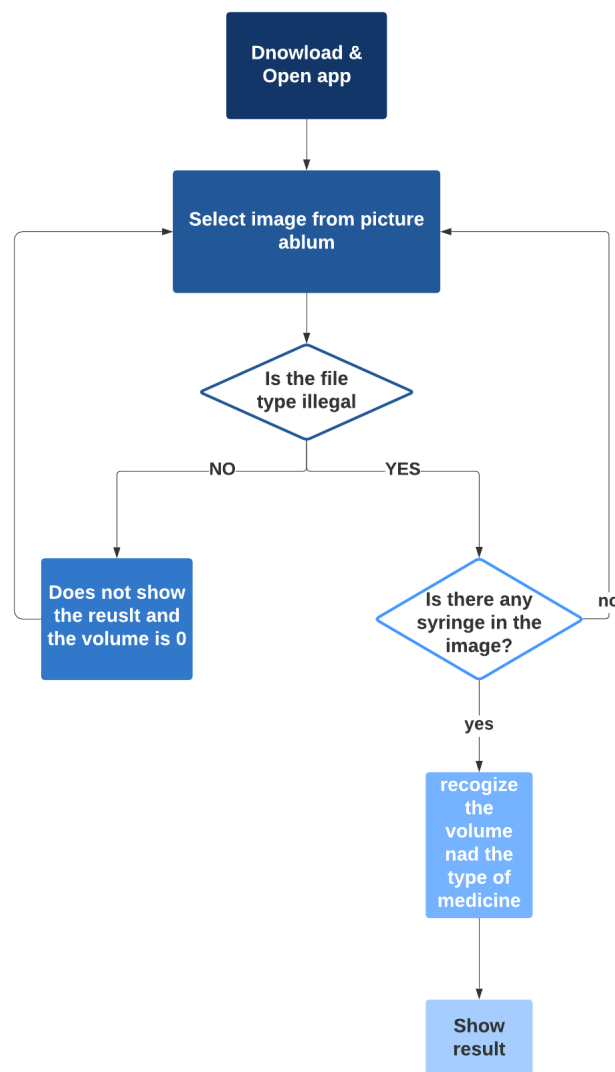


The work flow of the website is

1. User visit the website
2. If the user does not sign up, then let the user sign up. If the user has already signed up, let the user login.
3. User clicks the upload button and selects an image to upload.
4. The website recognizes the image by using core algorithms and does cross validation.
5. The website shows the result of recognition.

The whole workflow contains the request of the database to validate the username and password and the insert operation of the database. Also, it calls the APIs of the server and uses the calculation ability of the server. At the end, the result will be shown on the web front-end page. If the user uses some script to get the result, we also provide an API document to help them to create the request and get the result. The recognition limitation is linked with the account of each user. Also, the QPS of the common users is limited. These actions can help us to manage the website. All other requests will be denied by the server. We also set the identification of the account as the email. If the email address is abandoned, then the user can email us for further help and we will help them to recover the account and history images.

2.3.2. Workflow of Android request processing



The work flow of the website is

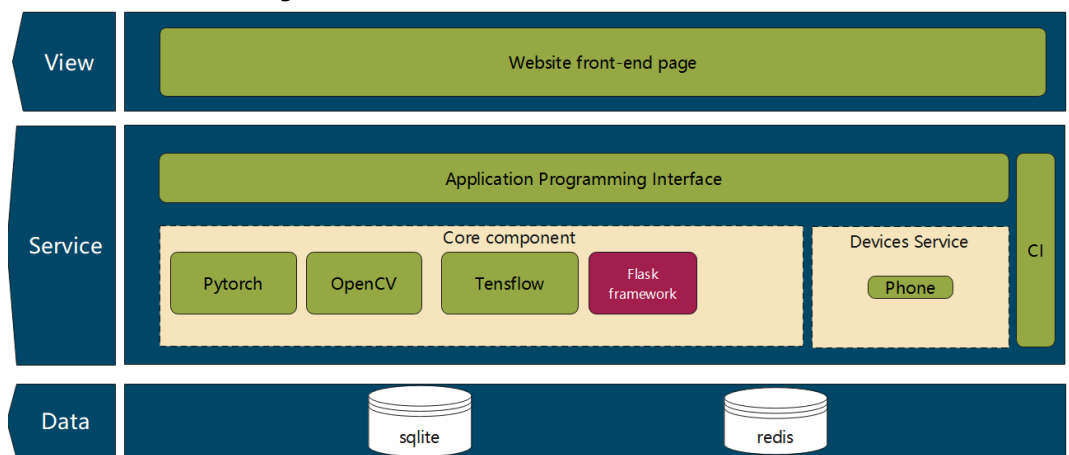
1. User download app and open the app

2. Select the image from the local album. If the type of file is illegal, for instance, the extension of the file is not .jpg or .png, then the program will give some text. If the image is legal, then the algorithm will start recognizing the images.
3. If there is a syringe in the image, then the core algorithm will recognize the type and volume of the medicine. If there is no syringe in the image, then the algorithm will return null and 0ml for the liquid medicine.
4. The user will see the result and re-choose images.

For the workflow of the Android app, it is much simpler than the workflow of a website. It only has one core algorithm and simple user interface. It offers an offline version of recognition. If users want to use script to batch process some sort of images, then the limitation is the computing power of the device. If users use an Android emulator on the computer, then they can batch process the images.

The validation of the app is not hard due to the fact that the app will mostly be used offline. There could be some type of private image format so that if OpenCV can recognize the image, then the format will be approved.

2.3.3. Overall web system architecture



SetDose System Structure

We divide the whole system into three parts: view layer, service layer and data layer.

- The view layer is the web front-end page. This layer conducts the direct communication with the users. All the login, uploading and showing the result is done in this layer. This layer is quite significant because this layer decides the user experience. If the view layer is not user friendly, the whole website will receive huge negative feedback.
- The Service layer. This layer contains three core algorithms and cross validation. It is the core layer of the website and

offers APIs. The Flask framework is the basement of the total website. Pytorch, OpenCV, Tensflow are three individual parts that can be separated. The flask works to offer APIs of the website and do QPS control, also with features such as user identification and search for historical images. The three core algorithms are fused into the website and they are in charge of recognition. If one of the algorithms needs to be updated, the other two can also do the recognition. The flask combines three core algorithms and forms the website.

Also, the Android app is on the service layer. The app offers service on the phone, which is different from the web service. The phone service will focus more on the main service rather than the expression. Thus, the main feature of the device service is the recognition of the images.

- The data layer. This layer includes two main algorithms: SQLite and Redis.

The SQLite contains two tables. The first one is the user records part. This table contains all user details including username, password, email, and other information. The unique identification of the user is the email. Also, we set the email column of the database to be unique. The other information is optimal. The other table is to store the uploaded images with the user id. When the user reviews the history images, the server will read the corresponding records and show the images on the web front-end page.

The Redis stores the uploaded images. After uploading the image, the server will convert the image to binary based on the base64 algorithm. It is a built-in algorithm of python. Then the Redis will store the image. Redis is a key-value no-SQL database. The key is image_id with the user_id. All keys in Redis are unique and if we want to read the image, we can find it directly. All keys do not have an expiration.

2.4. Technical Constraints

2.4.1. The constraints of the initial algorithm

2.4.1.1. Use machine learning model

As the size of our data set is limited to approximately one thousand images, and the accuracy of our model relies on the size of the data set, the accuracy might be influenced by our limited capability of gaining training data. For industry use, the dataset should be up to a billion. However, we only have about one thousand pieces of data. Both Pytorch and Tensflow will lose accuracy. We have used an alternative way to train the

model by using random forest. The way is to divide the whole dataset into several parts. We divide the whole dataset into ten pieces and nine of them are the training dataset, the left one is the test dataset. Due to the limitation of the size of the dataset, we decided to change the percentage from 9:1 to 8:2 and 7:3. By using this way, we can provide more training data and avoid overfitting in some way.

2.4.1.2. Recognize the volume

By analyzing the scale and the position of liquid This approach could work ideally with clean images. However, it might not be effective if the input images have too many unrelated interferences. There are two main difficulties:

2.4.1.2.1. the quality of images

The quality of images could be low or high depending on the quality of the camera and the background light. Also, the shooting angle could be also the main factor that affects the process of OpenCV. For this reason, we have added some text to notify users to use a high-quality camera, and use the flashlight. This situation occurs less in the hospital because, in the hospital, doctors and nurses have high-quality scanners and strong background light.

2.4.1.2.2. The syringe scale could be unclear.

The scale of the syringe may become unclear due to the fact that the syringe will be in use and disinfected by disinfection way including Ultraviolet rays and high temperature. These ways could make the scale hard to recognize so that OpenCV can not see the total volume of the syringe. Even though the separation of the rectangle(described in the OpenCV recognition part) could not be affected, the total accuracy will be much lower than normal.

2.4.1.2.3. The background-color

For now, the background color of images is pure color like green, red and white. These three colors are the common background color and are easy to recognize. If the background color is not pure color and hard to recognize, then the algorithm may lose control on writing the rectangle, so that the parameter in the formula could be wrong. In the worst case, the volume could be a negative number, and also the type of medicine will also be wrong.

2.4.2. The constraint of the dedicated server

As we decide to build a web application first which would require a server, the server might be restricted under our localhost. So we might have to rent a remote server, which may cause some limitations in performance and efficiency.

The public host is limited and the web cloud server is expensive. Thus, for now, we offer service for free and if the users reach some levels, users may have some limitations of uploading images and users will pay monthly.

Also, we have not tested the performance of the server in the situation that thousands of users visit the server at the same time. The reaction time could be longer and the availability of API could be lower. In the worst case, the server may shut down and refuse to offer service. Although we have set up the limitation of the API, the connection pool of database and memory will be used out. For this situation, we have advised the deployment device to be more powerful.

2.4.3. The constraint of testing different environments

As our resources and devices are very limited, the compatibility under different environments cannot be fully satisfied. There might be unpredictable issues due to this constraint.

For now, because the Android version is complex and phone products such as Sony, Huawei have their own UI and some of the low-level API of the operating system could be changed. These APIs are immutable while the app is running so that there may be some errors occurring and the app will exit. Also, the other significant thing is the SoC of the smartphones could be less powerful and not support the algorithm. This situation often occurs in old smartphones. We plan to do more tests on the Android emulator to identify more phones that can use this app.

2.4.4. The constraint of security

Due to the limitation of our knowledge and time, the security feature of our application may unavoidably have unexpected vulnerabilities. In more detail, there may be some attacks on our server, database, or code, but we may be short of technology and experience to defend them and fix the bugs.

For the security factor, we have set up the queries per second(QPS) to avoid the Dos attack. This setting is aimed at preventing the single user. If the attacker uses a DDoS attack, then the server may shut down.

Also, because we have a root user that can delete all data and manage the history data of other users, the attacker may use a dictionary to guess the username and password of the root user. If the password is covered by the dictionary of attack, then all user data may leak. The way to solve this is to increase the security level of the database and website. Because we have limited knowledge about security, we have not started the security test yet.

In the end, because we use sqlite, comparing the transitional database such as MySQL, it is much safer. However, because we store history images in Redis, the Redis server could be attacked. If Redis shuts down, then the history images preview function will not run and cause the shutting down of the server.

2.4.5. The constraints of usability

Due to our limited UI design experience and knowledge, our UI might not be as user-friendly as expected. The usability of our project might be influenced due to this limited UI design.

Even though we state that no design is the best design, if we have some knowledge or a UI/UX designer, we can set more hints to help users find corresponding buttons, and then the whole user experience could be more friendly. The same situation occurs in the Android app. Because we only set a clickable button and the user could only click that button. If we have a front-end engineer, we can use more complex components.

3. Quality of Work

3.1. Test plans:

3.1.1. Unit test for the method of traditional algorithm OpenCV for recognition

3.1.1.1. General description:

The most important function we need to realize in this project is uploading pictures and identifying the type and color of drugs in syringes in pictures, and the most important function behind this function is the recognition algorithm. Here we unit test the traditional OpenCV recognition algorithm

3.1.1.2. Tests that have been conducted:

3.1.1.2.1. Test for dataset to be trained:

Since we need to adjust the image size and color value from the original data set to improve the accuracy of the traditional algorithm, we need to check whether the image is suitable for testing. In detail, we need to convert the original image to a binary image containing only white and black and adjust the size appropriately to make the image sharper. After that, we checked whether the binary image was successfully converted and whether the image area was large enough.

3.1.1.2.2. Test for image reader:

This is a very important step. We tested whether the traditional algorithm OpenCV supported popular image types, such as .jpg, .png, etc. In addition, we tested the image reading part for errors. The read part is that the program will successfully read the image. If not, the program notifies the user with an error message.

3.1.1.2.3. Test for recognition rectangle:

Because we use a rectangle to frame the injector after the image transformation to get its estimated volume, it is important to enforce the correctness of the rectangle. We test whether the rectangle drawn by OpenCV covers all parts of the injector in the image.

3.1.1.2.4. Test for the volume estimation:

After drawing the rectangle, we draw a line across the piston of the injector in the picture. And then we can calculate the ratio of the non-liquid to the liquid, and then we can multiply the ratio of the non-liquid times the total volume to get the volume. This line is a critical estimate, so we tested to see if it is accurate on the edge of the piston, not on its middle line.

3.1.1.2.5. Test for color recognition:

The recognition of color determines the type of drug, so in order to identify drugs, it is important to test whether traditional algorithms can accurately identify colors. We have drawn the rectangles in the above test, and then we tested whether the traditional algorithm can successfully recognize the colors in the rectangles and can map these colors to the corresponding drug types. We used and verified images with different background colors, such as white and green. In addition, we added some mixed color background images as a difficult test.

3.1.1.2.6. Test for no injector images:

In the previous tests, we assumed that the image should contain an injector. If the user uploads an image without an injector, it may cause the program to crash. Therefore, we need to test whether the traditional algorithm can detect the error of no syringe in the recognition and output the corresponding error message.

3.1.1.2.7. Test for multi-injector:

Similar to the test picture with no registry mentioned above, we also tested the situation with multiple syringes in the picture. We test whether the program of the traditional algorithm can detect this condition and output the corresponding error message.

3.1.1.3. Test that will not be done:

3.1.1.3.1. Test for scalability:

Because our program is a single program, if we want to add more features, then we prefer to use the RPC framework to do that, rather than leave some components.

3.1.1.3.2. Test for recognition for different sizes of syringes:

Our program has assumed that the syringe size is fixed at 20ml, which is the result of our communication with the customer, because this is the typical size of the syringe used in the home, and the data set selected for the test meets this mentioned criteria. So we would not test whether the program can accurately identify syringes of different sizes.

3.1.2. Unit test for machine learning model:

3.1.2.1. General description:

For the function of the syringe in the recognition picture to be completed in the project, we not only adopted the traditional algorithm mentioned above but also tried to use the machine model to recognize the color of the drug. Specifically, we adopted two kinds of convolutional neural network models, one is the ready-made LeNet5 model and the other is the CNN model built by ourselves. Here we unit test two types of machine learning, including the pre-processing stage, the model itself, and the accuracy of the model. And since the process of model building and training is similar, Here I will talk about unit tests for them together.

3.1.2.2. Tests that have been conducted:

3.1.2.2.1. Test for dataset to be trained and tested:

Because we need to adjust and crop images from the original data set to improve the accuracy of the model, we need to check that the images in the training and test data set are what we want. To be more specific, we need to check the RGB channel, width, height, and other parameters of each image in the two data sets. The main reason we did this test was to check that the size of our data set was ideal.

3.1.2.2.2. Test for data loader:

While this is a very simple step, it can help us find and avoid many unnecessary errors in our code. For example, if we pass too much data to the program through multiple child processes at once, it is very likely that the program will crash, and testing the Data Loader can avoid this problem as much as possible.

3.1.2.2.3. Test for several functions in the model:

Because there are many convolution layers and pooling layers in the model, we will integrate these different layers into many functions and eventually run these functions to get the final result. We need to do this test to make sure that the relationship between all the inputs of these functions and the convolution layer is correct.

3.1.2.2.4. Test for the loss value:

Since we use the gradient descent loss function to basically evaluate the performance of different models, there is always a variable called loss value at each stage to measure the performance of the training, and we also need to adjust the parameters in the model according to the loss value. Therefore, we need to check whether the loss rate is decreasing, which means the performance of training is getting better.

3.1.2.2.5. Test for the accuracy of the model:

Because we want to use trained models to predict colors in syringes, we can automatically calculate the model's accuracy using only images from the test data set, which is the most direct and intuitive way to verify the model's performance. After testing, the testing accuracy is about 80 percent, which is not bad.

3.1.2.2.6. Test manually for accuracy of color prediction:

The reason for doing this test is also obvious. We can directly check whether the model correctly identifies the colors in the syringe by using new images from the validation data set. In the test, we selected 20 pictures from the test set and manually compared the prediction results of the model with the real colors. It was found that 15 pictures were correctly predicted, basically in line with the accuracy of the model tested above.

3.1.2.3. Test that will not be done:

3.1.2.3.1. Test for the performance on GPU:

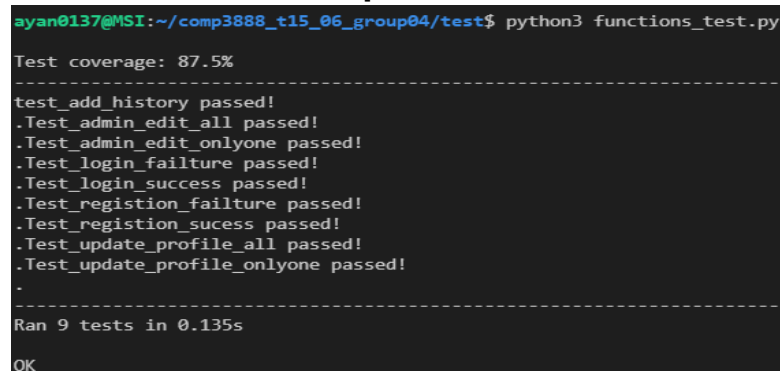
If the training data set is really large, the training speed will be very slow and it is best to move the model to the GPU. But in our example, the size of the dataset was acceptable, so we chose to train the model on the CPU by default. And according to our investigation, the performance of the model trained on CPU and GPU is basically the same, so we do not need to do this test.

3.1.2.3.2. Test for other complex models:

In this project, we chose to use the LeNet-5 model for machine learning to recognize different drug colors in syringes. Although the LeNet-5 model has been invented for a long time, its ability to solve simple problems is still relatively strong and efficient. However, other more complex models, such as AlexNet and VGGNet, are more capable of solving complex problems, but they are more likely to cause problems such as too long training time and insufficient efficiency. So instead of just having to recognize these three colors, the easy-to-use Lenet-5 model is enough. So, we don't need to test with more complex models.

3.1.3. Unit test for functions in Web application:

3.1.3.1. General description:



```
ayan0137@MSI:~/comp3888_t15_06_group04/test$ python3 functions_test.py
Test coverage: 87.5%
-----
test_add_history passed!
.Test_admin_edit_all passed!
.Test_admin_edit_onlyone passed!
.Test_login_failure passed!
.Test_login_success passed!
.Test_registion_failure passed!
.Test_registion_sucess passed!
.Test_update_profile_all passed!
.Test_update_profile_onlyone passed!
-----
Ran 9 tests in 0.135s
OK
```

Unit testing is defined as testing the smallest unit of a program. In the Web application, we use a variety of functions, such as registration, login, and change of profile functions, to help us more easily complete the Work of the Web application back end, so we can just do the smallest unit of the Web application back end of the unit test. For automated testing tools, we chose the Unittest package from the Python package, which is concise but very effective and provides a wealth of assertion methods, such as *AssertTrue()*, *AssertFalse()*, *AssertEqual()*, and so on. And can also automatically test and win test results. This is a tool made for web testing automation. After several rounds of testing, the coverage of the test example finally reached 87.5%, which proves that the majority of functions have been covered.

Here is the link of the test file: [Unit test for functions in web app](#)

3.1.3.2. Tests that have been conducted:

3.1.3.2.1. Tests for register function:

We implemented the registration function primarily to help the Web back end implement registered routes. Specifically, this function will first check that the mailbox to be registered is already in the database's user list, which means that the user has already been registered. If the user is already registered, a registration failure message is returned, and if not, the new user's email and password and other relevant information are injected into the database and a registration success message is returned. In this test, we mainly test these two cases, testing whether the registration function can correctly determine whether the email has been registered, and if not, whether the registration can continue successfully.

3.1.3.2.2. Tests for login function:

We implemented the logon function mainly to help the Web back end to implement the logon routes. This function checks if the email is already registered, and returns a message indicating that the login failed. Secondly, it will check whether the password entered by the user is the same as the corresponding password stored in the database. If it is not the same, it will return an error indicating that the login failed. This function also checks if the password entered by the user has been converted to the hash code for security purposes. And our test is mainly to detect the above three cases. The program can effectively detect the error, and if there is no exception, the program can normally help the user login, and return a successful login information prompt.

3.1.3.2.3. Tests for update profile function:

We implemented the change profile function mainly to help the Web back end to implement routes for users to change their profile. Specifically, this function locates entries in the database based on the user's email and then updates the database based on the information the user wants to change. Our test is to see if the function can let users change their profiles at will. At this point, however, we found a bug in our testing, which will be discussed in the next section. Finally, by fixing the bug, we realized the function that allows users to change all or part of their personal information and achieve the initial goal.

3.1.3.2.4. Tests for adding history function:

We implemented the function of adding history, mainly to help the Web back end to enable users to view history routes. Specifically, this function will create a new entry in the history table of the database based on the user's personal ID, which represents a new history record, and fills in relevant information such as the date, time, and identification result of the syringe at that time. This test is mainly to see whether the function can correctly write information into the history table and whether it can return the corresponding history ID.

3.1.3.3. Test that will not be done:

Tests for connecting with database: This item to the database correctly link test is very important, but we found that in the previous tests, both for landing, registration function test, or for change of personal information and add a history test, we test functions are established on the basis of the first database is the correct link. So in other words, the testing of the database has been thoroughly tested in the previous tests, including the deletion, addition, and change of entries in the various tables of the database. There is no need to do separate tests on the database.

3.1.3.4. Bugs found by testing and solutions:

3.1.3.4.1. Bugs:

We found a few bugs in our unit testing of various functions, the most typical of which was a bug in changing profiles. Specifically, before we designed this feature, we required users to be able to arbitrarily change the information in their profile, which meant that users could change all or just one piece of information in the profile. But when tested, earlier versions only allowed users to change all information, and when they changed just one, the program crashed with an error.

3.1.3.4.2. Solutions:

```
query = ""
if (username != " "):
    username = "Username = '" + username + "'"
    query = query + username + ","
if (email != ""):
    query = query + "Email = '" + email + "'"
if (age != ""):
    query = query + "Age = '" + age + "'"
if (address != ""):
    query = query + "Address = '" + address + "'"
if (gender != ""):
    query = query + "Gender = '" + gender + "'"
if (occupation != ""):
    query = query + "Occupation = '" + occupation + "'"
if (status != ""):
    query = query + "Status = '" + status + "'"
if (new_hash != ""):
    query = query + "Hash = '" + new_hash + "'"
if (query.strip() == ""):
    return True
elif (query.endswith(",")):
    query = query[:-1]
sql = """
UPDATE User
SET {query}
WHERE Email = '{cur_email}'
""".format(cur_email=cur_email, query=query)
```

To address this bug, we added a new feature in the function that allows unmodified information to remain unchanged. Specifically, if the user's input at a location is empty, meaning that he has not changed the information there, then the corresponding location in the SQL statement that performs the change of the database-related entry is also empty. This way, if the user changes only one piece of personal information, the profile page can still be updated successfully.

3.1.4. API test for the Web app:

3.1.4.1. General description:

```
ayan0137@MSI:~/comp3888_t15_06_group04/test$ python3 big\ api\ test.py
registerGETstatus:<Response [200]>
registerPOST NO.0register status:<Response [200]>
registerPOST NO.1register status:<Response [200]>
registerPOST NO.2register status:<Response [200]>
registerPOST NO.3register status:<Response [200]>
registerPOST NO.4register status:<Response [200]>
registerPOST NO.5register status:<Response [200]>
registerPOST NO.6register status:<Response [200]>
registerPOST NO.7register status:<Response [200]>
registerPOST NO.8register status:<Response [200]>
registerPOST NO.9register status:<Response [200]>
registerPOST NO.10register status:<Response [200]>
{'register test': [{'GET': <Response [200]>}, {'POST': <Response [200]>}]}
```

```
*****
PROFILES POST No2 14No of history status:<Response [200]>
PROFILES POST No2 15No of history status:<Response [200]>
PROFILES POST No2 16No of history status:<Response [200]>
PROFILES POST No2 17No of history status:<Response [200]>
PROFILES POST No2 18No of history status:<Response [200]>
PROFILES POST No2 19No of history status:<Response [200]>
PROFILES POST No2 20No of history status:<Response [200]>
PROFILES POST No2 21No of history status:<Response [500]>
PROFILES POST No2 22No of history status:<Response [200]>
PROFILES POST No2 23No of history status:<Response [200]>
PROFILES POST No2 24No of history status:<Response [200]>
PROFILES POST No2 25No of history status:<Response [200]>
```

```
*****
admin_page POST No10 time delete_id:0 status:<Response [200]>
admin_page POST No10 time delete_id:10 status:<Response [200]>
admin_page POST No10 time delete_id:21 status:<Response [200]>
admin_page POST No10 time delete_id:25 status:<Response [200]>
{'ADMIN_PAGE test': [{'GET': <Response [200]>}, {'POST': <Response [200]>}]}
*****
interface test coverage :99.89%
```

Because the Web application was the main component of our project, and the Web application had a lot of routes and interfaces, we tested these APIs using automated testing package Requests, and as shown in the figure above, we tested almost all of them. These include interfaces for registering, logging in, changing profiles, uploading images to identify results and history, and so on. Finally, after all the tests, we found that the coverage of the tests was almost 100%, which proved that all interfaces were detected, indicating the comprehensiveness and validity of the tests. In addition, we also consider the multi-user situation in the test of each interface and implement a certain degree of pressure test for the interface, which further proves the reliability of web applications.

Here is the link of test file: [API test for web app](#)

3.1.4.2. Tests that have been conducted:

3.1.4.2.1. Tests for the interface of signing up page:

First, we test the interface of the registration function at the beginning of the program. We split the test into two parts. First, we test the GET request method. We only need to make sure that the interface can return the registered template. Second, we tested POST requests to ensure that we could successfully pass in the registration information to the interface, such as email address and phone number. In order to simulate the scenario of simultaneous registration of multiple users, we conducted 11 cyclic pressure tests on the interface to ensure the stability of the interface in multiple consecutive registrations.

3.1.4.2.2. Tests for the interface of signing in page:

Similar to the above test for the interface of the registration function, we also divided the test for the interface of the login function into two parts. First, we test the GET request to see if the interface can return the login template correctly. Second, we test POST requests, which input registration information such as email address and password into the interface, and check whether the server can return 200, which means that it can successfully receive the user's request. We also conducted 11 cycles of pressure tests on the landing interface to verify its reliability.

3.1.4.2.3. Tests for the interface of upload images page:

Our interface for uploading pictures and identifying syringes is also divided into two parts. In the first part, as above, we test the GET request to see if the server can return the templates. The second part is a bit different from the previous one. Here we test the POST request, where the data sent to the interface becomes a picture rather than a simple message. Here we also check whether the server can return the 200 signal that represents a successful reception. Similarly, we perform 11 cycles of stress tests on the upload and recognition interface to test the reliability of image recognition when multiple users upload images continuously.

3.1.4.2.4. Tests for the interface of the profile page

We also tested the interface to view the profile page differently. Because viewing a profile page is more about getting data from the server and database, we'll mainly test GET requests here. We tested whether the server could pass back the corresponding templates and a set of relevant user profiles such as the user name and phone number through the interface. In addition, we also check the POST request to see if the server can correctly return signal 200. We also stress-tested the interface to view the profile page, simulating a multiplayer scenario, to ensure its reliability for continuous use.

3.1.4.2.5. Tests for the interface of editing profile page

Similar to the above tests, we first test the GET request of the interface for editing profiles to see if the server can correctly return templates for editing user profile pages. Second, we test the POST request, but we pass more data into the interface, such as name, gender, age, home address, and occupation, to see if the server can successfully accept that much more data, and return a 200 signal indicating success. We also simulated the scenario of multiple users using the function of editing personal information at the same time and conducted 11 cycles of stress tests to ensure the reliability of the function in multiple users.

3.1.4.2.6. Tests for the interface of history detail page

Similar to the previous interface test for viewing a user's profile page, we also focused on testing GET requests because the history page is also more about pulling data from the server and database. So we're more interested in seeing through the interface whether the server can return templates for the history details and related information such as the date, time, and the recognition result of the drug. In addition, we also simulated the situation of multiple users viewing the details of historical records to ensure the reliability and stability of this function under the pressure test.

3.1.4.2.7. Tests for the interface of the admin page

Interface testing for an administrator page is more like a combination of a profile page and an edit profile page. We need to test the GET request to see if the server can return templates and basic information for all users. The POST request is also tested to see if the server can return signal 200, which means it can successfully receive incoming data like the data information to be modified.

3.1.4.3. Tests that will not be done:

For API testing of Web applications, our goal was to test all interfaces successfully, and we did. This is because the complete API interface test can guarantee the reliability and stability of all lines and functions of The Web application to the greatest extent, which is the guarantee to ensure that our Web application becomes a high-performance application. So we did not abandon any aspect of testing in API interface testing.

3.1.4.4. Bugs found by testing and solution

3.1.4.4.1. Bugs:

We also found some web application bugs in our API testing. Here is a typical example. We in the historical record details page interface testing, found that all of the details page of history share the same routes and interface, which can lead to a problem is when the server wants to take the information of one of the historical records from corresponding entries in the database, it cannot get them success according to historical record ID. Because at this point, the server does not get a specific history ID through the interface.

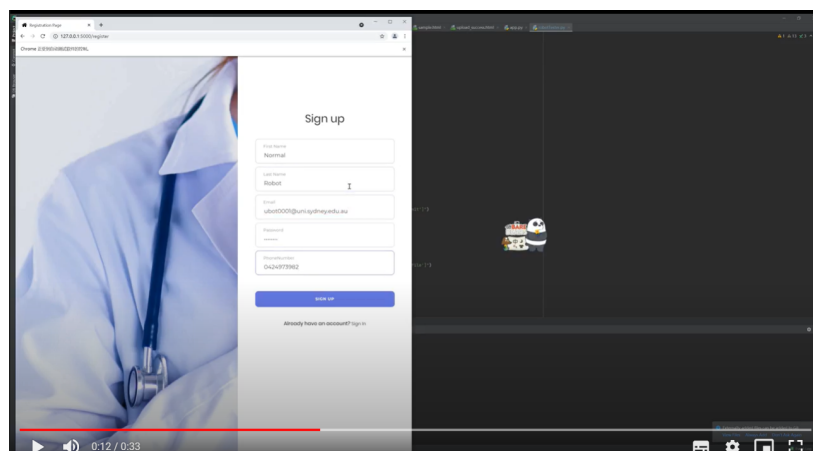
3.1.4.4.2. Solution:

```
@app.route('/history/<history_id>', methods=['POST', 'GET'])
def history_detail(history_id):
```

To solve the problem described above, we refer to the concept of dynamic routes, or dynamic interfaces, as shown in the figure. Specifically, we mark the interface of each history detail page with a history ID so that the server can search the database for the history information corresponding to this history based on this ID. In this way, each history detail page can accurately display the details of each history.

3.1.5. Regression test for the Web app:

3.1.5.1. General Description



[Demo video of regression test robot](#)

Regression testing is defined as retesting after changes to the old code to ensure that the original functionality of the program can remain intact after modification. For our Web application, we found a Web automation testing tool called Selenium to make it easier to automate regression testing. Using this tool, we designed a robot that can automatically use the functions of the website in the browser as shown in the above picture and demonstration video. The whole process is just like real human operation and the whole process is automatic. We use this robot as a validation tool for regression testing. If every time we added an important feature to the Web application or fixed an important bug, we would run through the robot to make sure that the functionality that existed before the change worked. At this time, if there are bugs, we can timely modify them, rather than wait until there are a lot of bugs piled up, which can ensure the continuous and stable progress of the project.

Here is the link of the test file: [Regression test for the web app](#)

3.1.5.2. Tests that have been conducted

3.1.5.2.1. Regression tests for signing up part

Throughout the regression test, we first focus on the test registration part. We use robots to simulate real people to fill in registration information, such as email, name, phone number, and password, in the Web application in the browser. There will be a proper pause of 1 second to give the robot time to react. We can monitor the various operations in the browser, and if we find that the final registration is successful, the regression testing of the registration part is complete, which means that the registration function can continue to be fully used after the addition of new features. We ran this regression test after adding new features such as login, uploading pictures, profile pages, and history details.

3.1.5.2.2. Regression tests for logging in part

After that, I'll introduce regression testing for the login section. Similar to the above mentioned, we can use the robot to fill in the email address and password required for login in the Web application in the browser like a real person and click the login button to complete login. If we observe the browser and find that the robot can successfully complete a series of operations to complete the login, then the regression test of the login part is complete. This also means that the login function will still work after the new function is added. We always do this after we add new features like uploading pictures, profile, history, and logout.

3.1.5.2.3. Regression tests for uploading images part

Here I describe regression testing for uploading photos and identifying results. This time, we will use the robot to upload sample photos in a Web application in the browser, simulate real users, and click the Upload button to complete the upload. At this time, if we can see in the browser that the robot can complete a series of operations to upload the picture, and the Web program can successfully return the recognition results such as the volume and color of the drug in the syringe in the picture, it means that the regression test of the uploaded picture has been completed. We always run this regression test after adding new features like profile, history, and logout.

3.1.5.2.4. Regression tests for profile part

Here I introduce regression testing for the profile section. We will use the robot to click the profile button in the Web application in the browser to enter the profile page and click the Edit personal information page to change some information at will and then click the submit button. When we observe in the browser, if we find that the robot can automatically and successfully complete these operations, not only can enter the personal information interface successfully, but also can change some personal information successfully, it means that the regression test of the personal information part has been completed. We did this regression test after adding new features such as history and logout, and after fixing major bugs.

3.1.5.2.5. Regression tests for the history part

Here we perform regression tests on historical detail records. We will use the robot mentioned above to click the detail button of a history item in the history overview of the Web application profile page to enter the history details page, and in the test, let the robot randomly click the delete button of a history item in the overview. We can check in the browser, see if the web application can be normal to jump to the record details page and check if the page can display properly before uploading pictures, testing the information such as date, time, and test results. And then we observe when the robot clicks the delete button, then a corresponding record is deleted. If all of these effects can be achieved successfully, then regression testing of the history is complete. After that, we do regression testing when we add new features or fix major bugs.

3.1.5.2.6. Regression tests for logging out part

Finally, we did a regression test for the logout function, which was a simpler step, using the robot to randomly click the logout button on several pages in the Web application. We then check to see if the Web application can jump directly to the login page after that, and if it does, it means that the retrospective testing of the logout

section is complete. After that, if we have a major bug fix, we do this regression test.

3.1.5.3. Tests that will not be done

We did not regression test the admin function because the admin section requires a special administrator account to log in and has its own special administrator center page, which is more like a separate section from the functions of ordinary users. And for the administrator part of the functions are relatively simple, just view the basic information of all users, change some of them and delete some illegal users. Therefore, after testing the administrator and confirming its normal operation, we separated it from the regression tests described above, which to some extent reduced the pressure of regression tests and improved the efficiency of tests.

3.1.5.4. Bugs found by testing and solutions

3.1.5.4.1. Bugs:

In our regression testing, we found many bugs in old features after adding new features or modifying major bugs. Here I will mainly introduce a typical example of a bug that bothered us for a long time and even took three of us three days to solve. Specifically, the recognition results page and the history details page in our Web application would normally display images that the user has just uploaded or uploaded before. At the same time, there is a problem in our Web application that UI files such as CSS files of registration and login page cannot be displayed properly. However, Xiaolin has solved this problem through his efforts. But the key problem is, after Xiaolin has changed the bug that CSS files are not displayed when we run the regression test again to make sure previous functions work well as normal, but we find that when we successfully get the recognition results and view the history details page, we can't view the images properly.

3.1.5.4.2. Solution:

The problem we discovered through regression testing seemed simple, but the process of solving it was really hard. At first, we thought there was something wrong with the HTML statement in Templates that makes the image display. However, after several attempts, we found that the bug was not caused by this, because the images would display correctly in local HTML files, but not in a running Web application. Later, we suspected that the folder where the CSS and other front-end files were saved had the same name as the folder where the images were saved, but the problem was not solved when we changed the name of the folder where we played. Xiaolin added a new line of code to the main program to make the CSS file display properly, and this new line of code caused the previous series of images not to display. We solved

the problem by putting this line of code in the right place and tweaking the folder structure.

3.1.6. Integration test for the Web app:

The definition of integration test is to combine all module units or functions into a completed system according to the requirements of structural design for integration test. Considering that our Web application didn't have a lot of blocks and features, just 7 or 8 major features, and the effectiveness of the combination of features was demonstrated in the various tests described above, we didn't plan to do overly complex integration tests. As it happens, the automated test robots created through Selenium used in the regression tests mentioned above are an excellent integration testing effort for simple Web applications. So when we finally tested whether all functions of the Web application worked properly together, we also chose to use the Selenium robot for integration testing.

3.1.7. System test:

The definition of system testing is that we take the software to be tested as an element of the entire computer system to see if it is compatible with the computer's hardware, peripherals, and other external environments. We didn't do specific system testing for this project, because the environment required for Web and Android apps is not too harsh. But while we were developing and conducting many of the tests described above, we were doing what amounted to simple system testing. For web applications, for example, we test that the application can only work properly when connected to the Internet, and run perfectly on Windows10, macOS, and other systems, which meets our initial requirements for web applications. As for the system test of the mobile app, we couldn't put it on The Apple Store because we didn't have qualified qualifications, so we didn't test the ios system. In addition, due to device limitations, we only tested the android App in the Android 8 system of Google Mobile 4, which only met our basic requirements for the android app applicable system.

3.1.8. Acceptance test:

I will discuss it in more detail in the next section "**Acceptance criteria and Acceptance tests**".

3.1.9. Limitations and constraints:

3.1.9.1. No tests for the real network communication with the server

Due to the limitation of capital and technology, we didn't really buy or rent a server like the Web application in the market. We just connected our Web application to the local server. So naturally, we couldn't test a network connection to a real remote server. There is also no way to really stress test web applications and

remote servers, to really simulate real multi-user scenarios, and to test the ability of web applications to cope with such scenarios. These tests are an important measure of whether a Web application is truly a high-performing application. So the lack of a truly remote server is still hindering our ability to fully develop web applications.

3.1.9.2. Not enough system tests for more different systems

In this project, we only conducted simple system tests with web applications and Android applications, especially for Android applications. As a result, we can only guarantee that the lightweight version of the Android app will run on Android 8 on the Pixel 4 phone, but not on many other Android phones and systems. If we had more time and developed on more diverse devices, such as Huawei, Samsung, and Xiaomi, we would have made Android apps more usable, allowing more users to reliably use our apps on their different Android devices.

3.1.9.3. The time of tasks in the test plan is not allocated reasonable

The comprehensiveness of the types of tests involved in our test plan is good, and basically, we have done all the tests needed in the project. However, due to the poor scheduling of the test plan, we did not make every test perfect. For example, in the first half of the project, we were so focused on the development task that we neglected to test it. As a result, we had a lot of testing pressure in the last few weeks. As a result, we didn't have enough time to perfect every test.

3.1.9.4. Not fully achieving the goal of Test-First development

Our goal was to implement Extreme Programming methodology, and an important part of that was test-First Development (TDD). As the name implies, test cases are first designed, and then developed based on test cases. However, we only carried out the development according to the guidance of the simple acceptance test, and more tests were carried out during the development process, rather than a series of specific test cases to guide the direction of development. Therefore, our test plan of this project can be said to practice part of the principle of the test first, but to a large extent, it did not fully achieve the goal.

3.2. Acceptance criteria and Acceptance tests

3.2.1. General description

Here I will introduce our acceptance criteria and acceptance tests for user stories. We will introduce the acceptance criteria and acceptance tests for each user story in three cases: normal, boundary, and abnormal. It is also important to note that the stories listed below are prioritized from most important to least important (the user story on the top is most important and the bottom is least important).

3.2.2. User story 1: As a general-public user, I want the software to check the syringe so that I can make sure I used the right dose

3.2.2.1. Normal:

- **Acceptance criteria:** the software should recognize the syringes with moderate dose inside in uploaded images and output the right volume of medicine in syringes.
- **Acceptance test:** We could upload a photo of a syringe with a moderate dose inside, see if the program can read his volume correctly.

3.2.2.2. Boundary:

- **Acceptance criteria:** The software should be able to recognize photos of syringes with very high or very low doses of the drug and output the correct volume.
- **Acceptance test:** We can check that the program correctly outputs the corresponding extra small and extra-large volumes by uploading a photo of a syringe with almost no drugs and a photo of a syringe full of drugs.

3.2.2.3. Abnormal:

- **Acceptance criteria:** The program should be able to recognize the image in which there is no drug in the syringe and output an error message.
- **Acceptance test:** We can upload a picture of an empty syringe and check if the program can detect that it is an invalid picture and output an error message.

3.2.3. User story 2: As a general-public user, I want the software to check the medicine inside the syringe so that I can make sure I used the right medicine

3.2.3.1. Normal:

- **Acceptance criteria:** The program should be able to recognize the photo of the drug in the syringe with a white background and output the correct type of drug in the syringe.
- **Acceptance test:** We can upload a picture of the drugs included in our program with a white background, and check whether the program can correctly output the types of drugs in the syringe.

3.2.3.2. Boundary:

- **Acceptance criteria:** The software should be able to recognize the picture of syringes filled with drugs with the background of other colors like green. And then it should output the right type of medicine.
- **Acceptance test:** We can upload a picture to the program with a green background, with a syringe filled with drugs, and check if it can output the correct drug information.

3.2.3.3. Abnormal:

- **Acceptance criteria:** If the drug in the syringe in the uploaded image does not belong to the type of drug detected by the program, the program should be able to identify the incorrect upload and identification and output an error message.
- **Acceptance test:** We can upload a picture, in which the types of drugs in the syringe belong to other than the three drugs included in the program, and check whether the software can find this and output the corresponding error information.

3.2.4. User story 3: As a general-public user, I want the software and server to recognize images as soon as possible so that I can get feedback quickly

3.2.4.1. Normal:

- **Acceptance criteria:** The software and server will return the result within 1 second.
- **Acceptance test:** Use a timer to record the time of identifying pictures, and perform this operation on enough pictures. Finally, calculate the average to judge whether the acceptance criteria are met.

3.2.4.2. Boundary:

- **Acceptance criteria:** The software and server will return the result within at most 2 seconds and at least 0.5 seconds.
- **Acceptance test:** During the above normal test, record the minimum and maximum time in the whole process.

3.2.4.3. Abnormal:

- **Acceptance criteria:** The software and server will return after 5 seconds if you upload images that do not meet the standard.
- **Acceptance test:** Inputting some nonstandard images to the server and see if the server can return after 5 seconds in these cases.

3.2.5. User story 4: As a general-public user, I would like to have an account to log in to the web application, for better security and more functions such as the history function

3.2.5.1. Normal:

- **Acceptance criteria:** Web applications should allow users to register with information such as email, name, and password, and have an account, as well as allow users to log in using this batch of information.

- **Acceptance test:** We can fill in various information according to the requirements of the Web application registration page, check whether the registration can be successful, and use the information used for registration can be successfully logged in.

3.2.5.2. Boundary:

- **Acceptance criteria:** Web applications should allow users to register multiple times with the same user name, phone number, and password, and log in based on that information.
- **Acceptance test:** We can register three to five accounts in the Web application, each with the same name and phone number, to check whether the registration can be successful, and check whether these accounts can be logged in successfully.

3.2.5.3. Abnormal:

- **Acceptance criteria:** Web applications do not allow users to register with the same email address, nor do they allow users to fill in the whole information required for registration. Also, web applications should reject login requests that do not match email addresses and passwords
- **Acceptance test:** We can register twice using the same email address to see if the Web application rejects us and prompts us. And when we enter the email and password do not match, check whether the Web application will reject the login request and give the corresponding prompt.

3.2.6. User story 5: As a general-public user, I want to have a secure personal user center with all kinds of information about me.

3.2.6.1. Normal:

- **Acceptance criteria:** The Web application should have a personal center that displays all kinds of personal information for the user, such as name, phone number, age, and so on, as well as an overview of the history.
- **Acceptance test:** We can access the profile center by logging into an existing account and checking the Profile button to see if the information is fully displayed.

3.2.6.2. Boundary:

- **Acceptance criteria:** Web applications should be able to display personal information in the personal center, even if it is a newly registered user, so for security reasons, the information option that is not filled in will be replaced by None.
- **Acceptance test:** We can create a new account and immediately log in and go to the profile center to check that it already displays all kinds of information, and the unfilled information option is None.

3.2.6.3. Abnormal:

- **Acceptance criteria:** Web applications should enable users with abnormal or irregular accounts to see their account status as abnormal
- **Acceptance test:** We can post some illegal information or upload some illegal pictures. After the account is banned by the

administrator, when we enter the personal data center, we can check whether the status of the account has changed to Abnormal.

3.2.7. User story 6: As a general-public user, I want to review the past record so that I can keep track and manage my dose

3.2.7.1. Normal:

- **Acceptance criteria:** The Web application should allow users to view the history of previously uploaded images and identification results, as well as the date, time, drug type, and volume of identification.
- **Acceptance test:** We can upload several photos of syringes first and get the identification results. After that, we can check whether we can see the history of these identification results on the personal center page, and click the details button to check whether we can enter the more detailed details page.

3.2.7.2. Boundary:

- **Acceptance criteria:** Web applications should allow users to view a large number of histories, such as 100 or 200 histories.
- **Acceptance test:** For our test, we can upload at least 100 pictures and get corresponding results, and then check whether we can see the overview of all history records in the personal center, and click into the details page respectively to check whether it can be displayed normally.

3.2.7.3. Abnormal:

- **Acceptance criteria:** Web applications should allow users to delete some history if they want to.
- **Acceptance test:** You can delete the history by clicking the Delete button in the History Overview, and then observe whether the deletion is successful and whether the deleted history is no longer in the history list.

3.2.8. User story 7: As a general-public user, I hope I can add and change personal information for privacy security and get advice from authorities

3.2.8.1. Normal:

- **Acceptance criteria:** Web applications should allow users to modify some of the information in their profiles
- **Acceptance test:** We can click the modify information button of the profile page to enter the modify page, and then modify the information of several items, and then return to the profile page to check whether the information has been updated successfully.

3.2.8.2. Boundary:

- **Acceptance criteria:** Web applications should allow users to modify all of their profiles, or just one of them.
- **Acceptance test:** We could do the same thing again, but this time modifies the entire profile or just one piece of information,

and then go back to the profile page to check that the changes were successful.

3.2.8.3. Abnormal:

- **Acceptance criteria:** Web applications should not allow users to change email addresses, because email addresses function as accounts in our applications and are the only credentials for users to log in.
- **Acceptance test:** We can repeat the above, but this time we change our email address and check if the program rejects our change and prompts us.

3.2.9. User story 8: As a general-public user, I want to use the recognition function more easily and conveniently anywhere.

3.2.9.1. Normal:

- **Acceptance criteria:** A lightweight version of an Android app should allow users to upload photos and identify them while offline
- **Acceptance test:** We can test whether we can use the uploading image and recognition function of the Android App in the offline environment.

3.2.9.2. Boundary:

- **Acceptance criteria:** Android apps should allow users to use the feature in more extreme situations, such as when the phone's battery is down to 10 percent.
- **Acceptance test:** We could get a couple of low-battery phones and test them to see if they can still use the uploading and recognition features in the Android app.

3.2.9.3. Abnormal:

- **Acceptance criteria:** Android apps should not allow users to use it in extreme environments, such as a phone with only 1% battery charge or running the app on a really old phone.
- **Acceptance test:** We can prepare some old Android phones and check if we can no longer use our Android app on them.

3.2.10. User story 9: As a general-public user, I want the software to be easy to use so that I can use it correctly

3.2.10.1. Normal:

- **Acceptance criteria:** Our Web app should allow users who have experience with normal Internet products to quickly become familiar with it.
- **Acceptance test:** We can invite five people outside of the project with general Internet experience to try out our Web application and gather their comments to see if they can quickly get familiar with how to use the application.

3.2.10.2. Boundary:

- **Acceptance criteria:** The Web application should allow people with little experience with the Internet to quickly become familiar with how to use it.

- **Acceptance test:** We can invite five people outside of the project with only a little Internet experience to try out our Web application and gather their comments to see if they can quickly get familiar with how to use the application.

3.2.10.3. Abnormal:

- **Acceptance criteria:** Web applications should be difficult to get used to quickly for people who have no experience with the Internet.
- **Acceptance test:** We can invite five people outside of the project with no Internet experience to try out our Web application and gather their comments to see if they cannot quickly get familiar with how to use the application.

3.2.11. User story 10: As an administrator, I want to manage all users and have the right to handle abnormal users.

3.2.11.1. Normal:

- **Acceptance criteria:** Web applications should allow administrators to view basic information about all users.
- **Acceptance test:** We can log in to the administrator account, go to the administration center page, and check to see if we can see the basic information of all the numbered users.

3.2.11.2. Boundary:

- **Acceptance criteria:** Web applications should allow administrators to change the status of users who violate rules or have abnormal accounts from normal to abnormal to remind them.
- **Acceptance test:** We can test if we can change the status of some of these users from normal to abnormal and back again in the administrative center.

3.2.11.3. Abnormal:

- **Acceptance criteria:** Web applications should allow administrators to delete zombie accounts in the administrator center that do not exist or are no longer used.
- **Acceptance test:** We can delete some test accounts from the administrator center page and check if they have been deleted from the user list. And we can also try to log in to the account that has been deleted to see if the program will tell us that the account does not exist.

3.2.12. User story 11: As a pharmacist, I want to use the software online for more accuracy.

3.2.12.1. Normal:

- **Acceptance criteria:** When working in an online model, the Web app can recognize the images.
- **Acceptance test:** Test the functions in an online environment and check whether the program can recognize the volume and colors of a bunch of images.

3.2.12.2. Boundary:

- **Acceptance criteria:** In the case of a poor network environment, the Web app can still recognize pictures.
- **Acceptance test:** We could use the 3G network to check whether the app can work normally.

3.2.12.3. Abnormal:

- **Acceptance criteria:** If the network is particularly poor, the app will stop using the Web app to identify pictures and switch to the local side and work as normal.
- **Acceptance test:** User a 2G network and check whether the Web app can work as normal.

3.2.13. User story 12: As a pharmacist, I want to use the software offline so that I can easily perform the measuring from anywhere

3.2.13.1. Normal:

- **Acceptance criteria:** When working in an offline environment, the Android app should display the correct volume, color, medicine name
- **Acceptance test:** Test the functions in an offline environment and check whether the Android app would be working as normal.

3.2.13.2. Boundary:

- **Acceptance criteria:** When working in a difficult and tough offline environment such as in the mountains with poor signal, the Android app should act like normal.
- **Acceptance test:** Test the functions in the place without signal to check whether the Android app would be working as normal.

3.2.13.3. Abnormal:

- **Acceptance criteria:** In an extremely difficult offline environment like the phone with 1% power, the Android app should give users a notification.
- **Acceptance test:** Using a phone with 1% power to run our Android app, and check whether it gives us a notification

3.2.14. User story 13: As a pharmacist, I want to read a volume from a variety of syringes so that I could use the software under more conditions

3.2.14.1. Normal:

- **Acceptance criteria:** Different syringe sizes can be recognized and volume can be calculated using different syringes.
- **Acceptance test:** Test for a bunch of different size syringes images, and check whether the volume in them can all be recognized.

3.2.14.2. Boundary:

- **Acceptance criteria:** The volume in the biggest and smallest syringes in the acceptance range such as 30 ml for the biggest and 10 ml for the smallest can be recognized.

- **Acceptance test:** We could upload several images of both the biggest and smallest syringes and check whether the program could check the volume in them.

3.2.14.3. Abnormal:

- **Acceptance criteria:** The program will notify the users if the size of syringes is too large or small.
- **Acceptance test:** We could upload larger syringes images like 100 ml size and smaller syringes images like 5 ml to the program, and check whether the program would give us a notification.

3.2.15. User story 14: As a pharmacist, I want to obtain the user's personal contact information with the permission of both the user and the administrator to provide professional medication advice

3.2.15.1. Normal:

- **Acceptance criteria:** Web applications should allow pharmacists to access users' email addresses and phone numbers with the administrator's permission to give advice to them.
- **Acceptance test:** We can simulate a pharmacist and check if we can get the user's contact information with the administrator's permission.

3.2.15.2. Boundary:

- **Acceptance criteria:** Web applications should allow pharmacists, with the consent of both administrators and users, to obtain more personal information about users, such as age and gender, in order to provide more professional advice to users.
- **Acceptance test:** We can impersonate the pharmacist and see if we can get more basic information about the user with two-factor authentication from the administrator and the user.

3.2.15.3. Abnormal:

- **Acceptance criteria:** Web applications should not allow pharmacists to access user information directly without the approval of administrators and users.
- **Acceptance test:** We can impersonate the pharmacist and check if the Web application will refuse our request for user information without the consent of the administrator and the user.

3.2.16. User story 15: As a data scientist, I want to use the website via the web browser.

- **Acceptance criteria:** The website will include completed web pages interfaces for them to use.
- **Acceptance test:** Act as a data scientist and use the web browser to go into the web application and check whether it could work as normal.

3.2.17. User story 16: As a data scientist, I want to use the offered APIs to recognize the images.

- **Acceptance criteria:** There will be comprehensive documents of APIs including request examples and response examples for them to take over the development work.
- **Acceptance test:** Go into the document or readme pages of the Web app and check whether there are comprehensive documents about API information.

3.3. Relevant testing techniques

3.3.1. Boundary analysis:

3.3.1.1. General description:

Boundary analysis is a very common software testing technique. The idea is to detect the input value of the program at the boundary. In this project, we mainly used this technique in the unit test of Web applications, which enables us to better test whether functions used in Web applications can still function properly when the input values are in the boundary state. Using this technique in unit testing of Web applications also allows us to better understand where the development process failed to meet the original goals so that we can improve the performance of web applications on a targeted basis.

3.3.1.2. Example 1:

```
def test_update_profile_onlyone(self):
    result = fc.update_profile("ayan0137@gmail.com", "", "", "", "",
                              "22", "", "", "", "")
    error_mess = "Test 6: test_update_profile_onlyone failed!"
    self.assertTrue(result, error_mess)
    print("Test_update_profile_onlyone passed!")
```

Here, we do the unit test for the function of changing profile information. Instead of changing several profiles, we tested whether the program would allow us to change just one profile. This is consistent with the idea of boundary analysis because we tested the minimal case of changing the profile section. After testing, we found that our program met this goal, giving users more freedom to change their personal information.

3.3.1.3. Example 2:

```
def test_update_profile_all(self):
    password = "1234567890"
    hash_code = hashlib.sha256(password.encode()).hexdigest()
    result = fc.update_profile("ayan0137@gmail.com", "", "An", "Yan", "ayan0137@gmail.com",
                              "22", "street No.1", "male", "user", hash_code)
    error_mess = "Test 5: test_update_profile_all failed!"
    self.assertTrue(result, error_mess)
    print("Test_update_profile_all passed!")
```

Here, we also do the unit test for the function of changing profile information. Unlike the tests mentioned above, we tested whether the program would allow us to change all the items in our profile if we wanted to. This also falls within the bounds analysis category, as we test the case of changing the maximum limit of the personal data section. After testing, we found that our program met this goal, which also gave users more freedom to change their profiles.

3.3.1.4. Example 3:

```
def test_registion_faillture(self):
    #this email has been already registered and we test if we could use it to sign up again
    actual_result = fc.register_user("test", "ayan0137@gmail.com", 480158532, "hash_code",
    | "normal", 0, "user")
    error_mess = "Test 1: test_registion_faillture failed!"
    # print(type(actual_result))
    self.assertFalse(actual_result, error_mess)
    print("Test_registion_faillture passed!")
```

Here we implemented unit tests on registered functions. Here we are testing whether we can successfully register with the email address even though it has already been registered and whether the program can detect the problem and give a hint. This test also conforms to boundary analysis because we test the extreme case of the registration scenario, which is to re-register using an already registered email address. After testing, we found that the program could detect the problem and refused our registration and gave us a prompt.

3.3.2. Stress test:

3.3.2.1. General description:

The definition of stress testing technology is to continuously put pressure on the program or server during the test, aiming to test the pressure tolerance of the program and ensure that the program can run under high intensity. In this project, we used this technique primarily for API testing of Web applications. Because our website has a wide audience, it is for all the children's families and medical workers, so it is inevitable that many people will use it at the same time or continuously. The API stress test can well simulate such a scenario of continuous use by multiple people, and can test whether our Web application can run normally under high pressure and complex environment, which can improve the reliability and stability of the Web application.

3.3.2.2. Example 1:

```
def login_page():
    get_data = {
        'data': 'get_login_page'
    }

    methods = ['GET', 'POST']
    results = []
    for method in methods:
        result = None
        if method == 'GET':
            result = send_get(login_url, get_data)
            print('loginGETstatus:%s' % (result))
        else:
            for i in range(11):
                firstname = str(i) * 8
                lastname = str(i) * 8
                email = str(i) * 8
                phonenumber = str(i) * 8
                password = str(i) * 8

                post_data = {
                    'data': 'post_register_page',
                    'firstname': firstname,
                    'lastname': lastname,
                    'email': email,
                    'phonenumber': phonenumber,
                    'password': password
                }

                result = send_post(login_url, post_data)
                print('loginPOST No%loginstatus:%s' % (i, result))
            results.append({method: result})
    return results
```

In this example, we tested the interface for the login section, the details of which are described in the test plan section above. But we should emphasize the for loop here, which simulates a scenario where multiple people log in in succession, which is likely to happen. So this stress test simulates the real world, putting constant pressure on the program and server to see if it can continue to run successfully under high levels of stress. After the test, we found that the interface of the login part passed the test and met our initial goal.

3.3.2.3. Example 2:

```
def upload_page():
    get_data = {
        'data': 'get_login_page'
    }

    methods = ['GET', 'POST']
    results = []
    for method in methods:
        result = None
        if method == 'GET':
            result = send_get(upload_url, get_data)
            print('testingupload GET')
        else:
            for i in range(11):
                firstname = str(i) * 8
                lastname = str(i) * 8
                email = str(i) * 8
                phonenumber = str(i) * 8
                password = str(i) * 8

                post_data = {
                    'data': 'post_register_page',
                    'firstname': firstname,
                    'lastname': lastname,
                    'email': email,
                    'phonenumber': phonenumber,
                    'password': password
                }

                login_result = send_post(login_url, post_data)
                print('UPLOAD-login POST No%login status:%s' % (i, login_result))

                file = ('test_file': ('1.png', open('/Users/zhiranbai/Desktop/main_code_test/static/upload_images/20190906_094830.jpg', 'rb'),
                    'image/png'))
                result = requests.post(url=upload_url,files=file)
            results.append({method: result})
    return results
```

In this example, we tested the interface for uploading images, as described in the test plan section above. We also emphasize the for loop here. This operation can simulate a scenario where multiple people upload pictures consecutively and obtain recognition results, which is also likely to happen because one or more parents may test several pictures of syringes consecutively.

So this stress test simulates the real world, putting constant pressure on the program and server to see if it can continue to run successfully under high levels of stress. After the test, we found that the interface of uploading and recognizing pictures passed the test, which proved the stability and reliability of uploading and recognizing pictures.

3.3.3. Equivalence partitioning:

3.3.3.1. General description:

Equivalence class partitioning is an important technique in black-box testing. Its specific meaning is that we divide the data or input fieldset into a part for testing with some standard, so as to deduce the test result on the whole data set. This can effectively avoid the problem of low test efficiency brought by the exhaustive method. In the project, we used this technique mainly in unit testing of machine learning.

3.3.3.2. Example:

As we have mentioned in the unit test of the machine learning model, we randomly divided the pictures into the data set into the training set and test set. We used the training set to train the model and the test set to test the performance of the model. To some extent, this also practices the concept of equivalence class division. Because we do not use the whole data set to test the accuracy of the model, but only select a part of it to evaluate the performance of the model, and then deduce that the model will have similar performance on all the data. This can not only improve the efficiency of the test but also avoid the problem that the accuracy of the test is not so accurate because the data used in the training and the data used in the test are crossed.

3.3.4. Automated testing script

3.3.4.1. General description:

Automated scripting is a common technique in Web application development and testing, and can often be defined as making a computer perform a series of operations automatically using pre-written programs. In this project, we applied this technique to regression testing. As detailed in the regression Testing section above, we used Selenium to design a robot that can automatically perform all the functions of a Web application and run the robot after each new feature is added or a major bug is fixed to verify that the original functions are working properly. And this automated test robot actually falls into the category of automated scripts.

3.3.4.2. Example:

```
def log_in(acc):
    email = browser.find_element_by_name("email")
    pwd = browser.find_element_by_name("password")
    email.send_keys(acc.get("email"))
    time.sleep(1)
    pwd.send_keys(acc.get("password"))
    time.sleep(1)
    button = browser.find_element_by_tag_name("button")
    if(button.text == "Login"):
        button.click()
    time.sleep(1)
```

As shown in the figure, we completed the automated script test for the login section in this example. We let the script automatically enter your email address and password on the login page and click the login button to complete the login. And the script waits one second in between to give the computer enough time to complete the operation. All we need to do is open the browser and watch the script complete the login function. If it successfully logs in, it means that the regression test for the login part is complete.

3.3.5. Use case testing:

3.3.5.1. General description:

In simple terms, use case testing is the use of test examples designed based on user behaviour to test whether the software can meet the needs of real users. This is exactly the kind of technology we can apply to acceptance testing in our project. A lot of the testing we do assumes that we are the user and checks to see if the functionality of the test satisfies the user story. In some cases, we invite people outside of the project directly to simulate real users, so as to make the application really fit the needs of users.

3.3.5.2. Example:

For example, use-case testing is clearly used in acceptance testing of software usability. In this test, in order to test whether the software is really easy to use, we invited five people outside the project to test, and they played the role of very close to the real users. After testing, we collect their comments and use them to determine if the user story is satisfied and further improve the ease of use and usability of the application's features.

3.4. Discipline knowledge

3.4.1. Quality constraints for software design

This project's team members are from the discipline background of computer Science major or advanced computing major. These two disciplines have prepared us with techniques, mindset, and guides during the design, development, and testing phase of this project.

For quality constraints we considered software design includes: security, performance, reliability, and availability.

For security, because the intention is to design an application that can run both online and offline, the security part we considered is mainly about server attacks. The server attack may include but is not limited to (SQL injection, Unauthorized access).

For SQL injection, with user input channels being the main vector for such attacks, the best approach is controlling and vetting user input to watch for attack patterns. So for the login page, registration page, and profile editing page, we have set format constraints on the input section. For example, in the phone number input, we required exactly 10 numbers for that section, the user will not be able to register below or above the 10 number limit.

For unauthorized access, we have defined user account authority. There are 2 types of accounts for our website, there is a normal user account for managing the users' own using history and there is an administrator account to manage all the user accounts. All the account login information is encrypted using hash functions, So it is safe to avoid unauthorized access. We considered these risks and implemented a protection mechanism on the server and code to reduce the attack risk.

About performance, for the colour recognition part, we have tried to use consecutive "for loops" and visualization toolkits "tensor board" to build the neural network structure in order to find the best neural network with minimal layers and maximal accuracy. To explain this in detail, the optimal convolutional neural network structure varies from question to question, so although the type of layers is fixed (convolution layer, max-pooling layer, flatten layer, dense layer) we still need experiments to find the best structure. Using the for loop we have tried out various layer structures and finally, we have a network structure with the highest accuracy.

For reliability and availability, we kept these 2 aspects in mind during the process of construction, combination, correction, preparation, and testing, each phase includes peer review, development plan before code is written, task-oriented development with extra care with the flask and android part of the development to ensure the software is failure-free and works properly under the desired environment.

Also, we run regression tests at least once every week. In this way we gain valuable experience on how our application operates, we can know what caused the problem and we can quickly fix it. In addition to that, we have tried to inject specific failures into our website, So that we can see how the website responds and implement edge case handling if necessary.

3.4.2. Research and use of relevant algorithms and design patterns to solve problems

For research and relevant algorithms, we have first researched several existing machine-learning projects, and from them, we gained basic insights into machine learning and decided to use open-source machine learning platforms like Tensor flow, Pytorch, and other tool kits like OpenCV and flask to build the recognition software for online and offline use.

The design pattern of a neural network is fairly standard, all neural networks consist of convention layer, max-pooling layer, flatten layer, dense layer. The difference is only the number of each layer. So as I mentioned above, we used test toolkits and visualization tools to measure and adjust the structure of the neural network.

The design pattern of the flask also follows a template. We use the “app. route” function to handle the backend of each page and “app. run” to start the application. For error handling, we used the “app. error handler” to detect and handle HTML status code.

The design pattern of the android follows the standard of API 26, which can cover the version of the Android app from 4.0 to 12. The app is backward compatible, we use the same OpenCV method and structure as the backend of the android application.

3.4.3. Complexity analysis of algorithms

For complexity analysis, we applied the knowledge from comp2123 and comp3027 to analyze, calculate and improve our algorithm.

The part worth mentioning is the convolutional neural network model chosen for the PyTorch approach. For PyTorch, we used the LeNet5 model which is one of the oldest convolutional neural network machine learning models. We have looked at other models like VGG16, AlexNet, ZFNet, and googLeNet but the reason why we finally chose LeNet5 was that it is famous for recognizing images.

Its features include: each convolutional layer always includes three parts: convolution, pooling, and nonlinear activation functions and it uses convolution to extract spatial features. It does sub-sampling on the pooling layer and uses MLP as the last classifier. It also has sparse connections between layers to reduce the complexity of computation. Although it is very simple, it has high accuracy and high training efficiency for the situation where we need to classify only three colours. In order to make it more convenient and integrated, several convolutional layers and pooling layers form a forward function and superimpose several such functions in the model to obtain the final model.

3.4.4. Technical details and knowledge relevant to Unicode

For TensorFlow, we first pre-process the data, including re-changing the size of the picture, data augmentation and standardizing the pixels in the picture, and so on. These data pre-treatment technology can well adapt to let the data and model, and the standardization of the pixels in the image and enhance data can increase the richness of data sets, let model come into contact with more abundant data, which can make a model deal with more complex scenarios, and avoid overfitting phenomenon of the model.

Secondly, we label the pre-processed image and put these images into a data bundle using “pickle”.

Thirdly, we set up the convolutional neural network model. It includes several convolutional layers, pooling layers, flattened layers, and dense layers to form a model. And used the tensor board to find the best network configuration.

Fourth, we used the “model.fit” function to run the image bundle with our neural network. Finally, the accuracy of these models with different parameters was tested with the tensorboard to evaluate the accuracy of different models, and the model with the highest accuracy was selected as the best model, which was saved to predict the colour and type of drugs in the syringe.

For PyTorch, We first preprocess the data, including re-changing the size of the picture, data augmentation and standardizing the pixels in the picture, and so on. These data pretreatment technology can well adapt to let the data and model, and the standardization of the pixels in the image and enhance data can increase the richness of data sets, let model come into contact with more abundant data, which can make a model deal with more complex scenarios, and avoid overfitting phenomenon of the model.

Secondly, we divide the pre-processed data into the training sets and test sets. The training set is used for training models, and the test machine is used for evaluating models.

Thirdly, we set up the LeNet5 model, which is one of the oldest convolutional neural network machine learning models. Although it is very simple, it has high accuracy and high training efficiency for the situation where we need to classify only three colours. In order to make it more convenient and integrated, several convolutional layers and pooling layers form a forward function and superimpose several such functions in the model to obtain the final model

Fourth, we use the training set that has been segmented above to train the model. Among them, we can adjust the learning rate and epoch in model training, because different learning rates, batch sizes, and epochs will affect the training efficiency and accuracy of model training. Generally, the higher the learning rate, the faster the convergence speed, but at the same time, it is easy to produce an over-fitting phenomenon. However, if the learning rate is too small, the over-fitting phenomenon will be avoided, but the convergence speed will be slow, and the optimal effect may not be achieved in the end. The size of batch size will affect the accuracy of training, but generally speaking, the smaller the batch size will make the model more accurate. Finally, the amount of epoch also determines the performance of the model. Too little epoch will make the model stop training before it is trained to the best performance, while too much epoch will prolong the training process and the model is prone to overfitting.

Finally, The accuracy of these models with different parameters was tested on the test set to evaluate the expressiveness of different models, and the model with the highest accuracy was selected as the best model, which was saved to predict the colour and type of drugs in the syringe.

The unit code relevant to the project includes comp2123, comp3027, info1110, info1113, isys2120, info2222.

4. Quality of Group Processes

4.1. The setup of tooling for development, management of tasks, allocation of tasks

4.1.1. The setup of tooling for development

4.1.1.1. Setup of Colour recognition algorithm

In the first half of the semester, our team members mainly focused on implementing the color recognition algorithm. After our communication with clients and the decision of our group, we decided to use Pytorch and TensorFlow from Python to recognize the color of the medicine, and OpenCV from Python to recognize the volume of the medicine.

4.1.1.1.1. Python programming: Vscode and Pycharm

We use python as our main programming language because it is easy to write and we all are familiar with it. And Pycharm is the IDE most of us are using for python programming.

4.1.1.1.2. Machine learning algorithm: TensorFlow and Pytorch

We used TensorFlow and Pytorch to develop our machine learning model for the medicine type and volume recognition.

4.1.1.1.3. OpenCV algorithm: OpenCv package

OpenCV is another approach we use to achieve the same goal as machine learning, which is an alternative for machine learning methods. We decide to experiment with all of them in our algorithm and apply the result with the best performance with cross-validation.

4.1.1.2. Setup of Website development

In the second half of the semester, we mainly focus on Website development and APP development. In terms of web development, we use SQLite as our general database, use Redis as our image database, and use flask as our website framework. For Android APP development, we use Java to program.

4.1.1.2.1. Android programming: Java 1.8

We use Java 1.8 for the development of our Android application aiming to support as many devices as possible.

4.1.1.2.2. General database: SQLite

We use the SQLite database to store all the user accounts and upload histories apart from the image file. We chose SQLite mainly because of its light feature and easy management. However, we would consider using MySQL if we are going to implement multi-threading for our application.

4.1.1.2.3. Image database: Redis

To store all the image files, we chose the Redis database because of its splendid performance for storing large files compared to SQLite.

4.1.1.2.4. Website framework: Flask package

We use flask as the framework of our website, the reason is just the same as why we chose SQLite, due to lightness and ease of use. Additionally, also because it is easy to extend which satisfies our frequently changing requirements.

4.1.2. Management of tasks

4.1.2.1. Weekly tasks management

For our weekly tasks management, we usually discuss the division of tasks on the Tuesday tutorial. After each team member confirms and understands their responsibility for the development work of that week, we will use Jira Issue for formal task division and set the status as Todo. In addition, team members need to push their codes to Commits after they complete their own tasks so that other members can view the progress. On Friday, after the client meeting is over, we will continue to talk about our development work progress and the group members will check the status of the task through Jira Issue. If the status of the task is not Done, corresponding members need to explain the reason or problem they faced.

Link to part of Jira Issue screenshot:

[screenshot1 of Jira Issue](#)

[screenshot2 of Jira Issue](#)

4.1.2.2. Branches tasks management

For stage development task management, we use Branches to handle it. In the first half of the semester, our main development tasks were divided into OpenCV, Pytorch, and Tensorflow

algorithm development, so we created three different branches, and uploaded all the code corresponding to that branch. However, in the second half of the semester, our main development work is about website application, since web development involves too much content, such as web front-end, website back-end, and database, all our codes are updated on the branch called the Flask.

Link to Branches screenshot: [Branches screenshot](#)

4.1.3. Allocation of tasks

4.1.3.1. Weekly status report and meeting minutes:

Xiaolin Zhao

Project status reports should be constructed every week.
There should be meeting minutes for every meeting we hold.

4.1.3.2. Communication with clients and client meeting minutes: An Yan

Communication with clients includes mostly email contact with clients, and the person who takes this task should also take more responsibility for communicating with clients in the meetings.

4.1.3.3. Wiki page management: Hurui Yang

The wiki page manager should manage the wiki repository and also the home page markdown file, which includes adding the hyperlink to the important documents on the wiki home page.

4.1.3.4. Scope and user story:

4.1.3.4.1. Scope: An Yan

The scope statement has been determined based on all the team members' discussions. However, An Yan is responsible for its final construction and management.

4.1.3.4.2. User story: Zhiran Bai

The user story has been determined and regressively updated based on all the team members' discussions. Zhiran Bai is responsible for its documentation and management.

4.1.3.5. Recognition algorithm:

4.1.3.5.1. Traditional algorithm with OpenCV: Hurui Yang

Traditional algorithm with OpenCV is the use of the OpenCV method to detect syringe's medicine type and volume which is likely to be based on the colour and scale from the image.

4.1.3.5.2. Machine learning using Tensorflow: Zhiran Bai

This is the machine learning algorithm aiming to determine the volume of the dose resulting from a trained model. Zhiran Bai's responsibility is to train the model using the dataset provided and adjust the model.

4.1.3.5.3. Machine learning using Pytorch: An Yan

This is the machine learning algorithm aiming to determine the medicine type of the dose resulting from a trained model. An Yan's responsibility is to train the model using the dataset provided and adjust the model.

4.1.3.6. Website front-end development: All team members

The front-end development includes mainly constructing the templates of the website. All the team members have a decent contribution to this task.

4.1.3.7. Website back-end development:

Our team completed the development of the back-end of the website in week8 and week9 such as ensuring that the website can connect to Redis and the website can correctly read the images of the specified corresponding path. After that, we focus on bug fixing of the website in week10.

4.1.3.7.1. Upload and recognition result routes: Hurui Yang

4.1.3.7.2. Profile routes: Zhiran Bai

4.1.3.7.3. Edit profile routes: Zhiyang Zhang

4.1.3.7.4. Register, login, history, and admin routes: An Yan

4.1.3.7.5. Database development and management: Zhiyang Zhang

4.1.3.7.6. Redis database management: Hurui Yang

4.1.3.7.7. Register, login bug fixing: Xiaolin Zhao

4.1.3.8. Android app development: Hurui Yang

The Android APP development was done by the Head Programmer of our team, Hurui Yang. He used Java to program an App that can read the medicine photos taken by the phone camera and return the volume and colour of the medicine to the users.

4.1.3.9. Test:

In order to meet the requirements of the Final Demo Presentation, we did five different tests. The test objects were machine learning models and web apps, but our main test object was still web apps. We used API test, Unit test, Regression test, and Acceptance test for our web applications.

- 4.1.3.9.1. Unit test for machine learning model: An Yan, Zhiran Bai
- 4.1.3.9.2. Unit test for functions in the web app: An Yan
- 4.1.3.9.3. API test for web app: Zhiran Bai
- 4.1.3.9.4. Regression test for the web app: Zhiyang Zhang
- 4.1.3.9.5. Acceptance test: Xiaolin Zhao

4.2. Evidence of collaboration and teamwork

4.2.1. Group roles

- **Manager:** Xiaolin Zhao
- **Tracker:** Xiaolin Zhao
- **Customer:** An Yan
- **Tester:** Zhiyang Zhang
- **Doomsayer:** Zhiran Bai
- **Head programmer:** Hurui Yang
- **Programmer team:** All members

4.2.2. Sharing of work and collaboration

4.2.2.1. Weekly collaboration

After Tuesday's tutorial meeting, we will discuss our development work arrangement for that week. Everyone's development workload is different according to their XP roles. Generally, there will be fewer tasks assigned to the Customer and Manager, because they usually have got more XP role work than others. Also, after Friday's client meeting, we will continue to have a meeting to discuss everyone's development work progress and feedback. This is how we collaborate together every week.

4.2.2.2. Teamwork collaboration

In week7 and week13, when we have teamwork such as Final/First Demo presentation, Final/First report, and Final/First Deployment, we will assign work to every team member at first. Then we will hold group meetings every day to confirm group members' daily work progress through the Zoom contribution screen and solve some problems and challenges through communication and collaboration. In addition, we often have four or five presentation rehearsals before the Deployment and Demo presentation. We will repeat the rehearsals until everyone can make the presentation smoothly. Manager XiaolinZhao will also time our speech to ensure that our speech will not be over time.

4.2.3. Group contract

Following the guidance of group contract:

All the group members have participated punctually in all the meetings including weekly tutorial meetings, client meetings, regular meetings, and occasional arranged special meetings.

All the group members have made rational contributions in programming, documentation, and presentation.

All the group members have been enthusiastic about helping each other. We have all been willing to help with others' problems that are in their unfamiliar area. Everyone feels supported by other teammates.

4.2.4. Meeting minutes

We routinely have meetings every week, though there are sometimes special occasions causing clients unable to attend.

However, there are special additional meetings that are not recorded, like meetings for report writing or presentation preparation, this might be improvable in the future.

- Week2: [client meeting](#), [tutorial meeting](#), [group meeting](#)
- Week3: [client meeting](#), [tutorial meeting](#)
- Week4: [client meeting](#), [tutorial meeting](#), [group meeting](#)
- Week5: [client meeting](#), [tutorial meeting](#), [group meeting](#)
- Week6: [client meeting](#), [group meeting](#)
- Week7: [tutorial meeting](#), [group meeting](#)
- Week8: [client meeting](#), [tutorial meeting](#), [group meeting](#)
- Week9: [client meeting](#), [tutorial meeting](#), [group meeting](#)
- Week10: [client meeting](#), [tutorial meeting](#), [group meeting](#)
- Week11: [client meeting](#), [tutorial meeting](#), [group meeting](#)
- Week12: [client meeting](#), [tutorial meeting](#), [group meeting](#)
- Week13: [client meeting](#), [group meeting](#)

4.2.5. Project status report

Tracker XiaolinZhao is responsible for writing the weekly status report. He usually completes the report on Saturday. Before submitting the report to Canvas, he will post the report in Slack or Wechat for the rest of the team members to confirm and upload it again.

- [Week3](#)
- [Week4](#)
- [Week5](#)
- [Week6](#)
- [Week7](#)
- [Week8](#)
- [Week9](#)
- [Week10](#)
- [Week11](#)
- [Week12](#)
- [week13](#)

4.3. Allocation of group roles, potential risks, constraints

4.3.1. Allocation of group roles

The roles of our team members were decided in week 2 and have not changed since then. It is worth mentioning that our development work will be determined according to each member's XP role. For example, XP roles such as Consumer and Manager usually have a relatively large XP workload, so the development work allocated to them will be relatively fewer.

- **Manager:** Xiaolin Zhao
- **Tracker:** Xiaolin Zhao
- **Customer:** An Yan
- **Tester:** Zhiyang Zhang
- **Doomsayer:** Zhiran Bai
- **Head programmer:** Hurui Yang
- **Programmer team:** All members

4.3.2. Potential risks.

Our risk register could be accessed from this [link](#). And here is the screenshot of the risk register

Risk No.	Risk name	Risk Description	Risk Owner	Category	Mitigation Plan	Impact level	Description of impact	Likelihood of occurrence	Contingency Plan
R01	Security Risk	Due to the limited time and resources, we may have security risks. Our website currently does not have the ability to prevent many common attacks like CSRF, DDOS etc. This means our website may encounter unpredictable issues under risks of being attacked.	All	Technology risk	Using different methods to prevent different attacks	medium	Influence on website operation. May even impact our users' privacy	<30%	Prolong the process duration
R02	Performance risk	Due to the limited amount of data, the performance of our model may not be as good as our own observation. This is likely to result from our limited data variety. The model might have unexpected performance due to various upload images with unpredictable features.	Zhiran Bai and An Yan	Technology risk	Obtain more dataset, download from online source or create by ourselves	medium	Influence on the model accuracy	>40%	Take photos of syringes containing medicine.
R03	Environment risk	Due to the limitation of our own equipment, we cannot guarantee that our application could be compatible in all of the running environments. Nevertheless	All	Environment risk	Use device environment simulator to simulate environments as many as possible.	low	May impact the usability	>40%	Prolong the process duration and increase budget
R04	Language risk	As our application has English as the only available language, there is risk that people who does not speak English cannot use our application.	All	Locality risk	Make our application to support more language as possible	low	Impact usability	>50%	Prolong the process duration
R05	Edge case risk	As our application has not been fully tested, there might be edge cases we have not discovered.	Zhiyang Zhang	Technology risk	Conduct as many tests as possible. / Publish a beta version first for testing.	medium	Impact usability and possibly also security	>40%	Prolong the process duration

Currently, it only has five risks that we could foresee for now due to the time limitation. It contains all the information including the impact level, occurrence possibility, and our mitigation plans about the risks, and it could always be updated in the future.

4.3.3. Constraints

4.3.3.1. Communication constraints

Since our communication happens all remotely online, it is unavoidably less efficient than face-to-face communication. Therefore, it usually takes more time to exchange information and knowledge, which undoubtedly constraints our working efficiency.

In order to best address this impact, we agree to reply to others' messages as soon as possible. And we often arrange small quick meetings when there is a problem that is hard to discuss with text only. Additionally, for achieving the best simulation of face-to-face communication, we agree to turn on our cameras if possible.

4.3.3.2. Language constraints

As all of our team speaks Chinese as our first language and we tend to use it for our communication. This is causing two problems, one is that we can hardly provide as much chat history as to how much we have talked. The second problem is that our communication with clients might not be as efficient as native English speakers. Though we all have tried to overcome this challenge as much as we could, this is a long-term goal to accomplish.

4.3.3.3. Time difference constraints

As three of our team members have been in China and two have been in Sydney, there is a three-hour time difference between us. Though it is only three hours, it increases the difficulty in managing our meeting time and our intensive communication time, because our available time is intensively 8 pm to 11 pm at night. We usually can only meet after China's members have dinner and before Sydney's members go to bed.

4.4. Use of bitbucket and slack and other tools

4.4.1. Communication: Slack and Wechat and Zoom

We used slack and WeChat for our textual communication and zoom for the live meeting. Slack is mainly for formal tasks discussion and WeChat is mainly for general chat. Additionally, we also set up a webhook from bitbucket to slack which informs the channel when someone made a commit in the bitbucket.

4.4.2. Document management: Bitbucket Wiki and Google doc

Bitbucket wiki is what we used to store all the textual work and evidence of what we have done. It includes all the meeting minutes, status reports, individual reports, database information, etc. The most important files' link can be found on the Wiki home page.

As for Google Docs, we mainly use it for constructing group reports and presentation slides together. And also due to its sharing feature, occasionally we would use it as a tool to edit a short snippet within the team.

4.4.3. Tasks allocation: Jira issue

We use the Jira issue to initialize and track all the tasks we have come up with. Routinely, we construct five Jira issues per week and all the team members would be assigned one issue. Then in order to track the status of the issue, it would be set "finish" when it has been done. Normally the weekly issues should be done within one week, the assigned member should explain when it is not.

4.5. Work with client

4.5.1. Weekly client interaction

4.5.1.1. Week2 client meeting

Link to week2 meeting minutes:[week2 meeting](#)

In week2, we asked clients about what aim we should achieve in this project and what kind of support we can get from clients. We understood the general requirements of customers for us after this meeting, and we need to decide how we are going to implement SetDose development before the next meeting.

4.5.1.2. Week3 client meeting

Link to week3 meeting minutes:[week3 meeting](#)

In week3, we reported to the clients that we decided that the function of the software should aim at recognizing the colour and volume of the medicine. Also, clients told us we can apply for financial assistance, for example, we can buy a lightbox for our medicine recognition. We will decide whether it is necessary to buy a lightbox.

4.5.1.3. Week4 client meeting

Link to week4 meeting minutes:[week4 meeting](#)

In week4, we showed our preliminary medicine recognition algorithm to our clients. We could recognize the context on syringes and transfer the images to a binary image, which is ready to do the rest of the work recognizing the volume in the syringe, clients were satisfied with our work. And we were trying to do machine learning of recognizing the colour in syringes in two ways, which are Pytorch and Tensorflow.

4.5.1.4. Week5 client meeting

Link to week5 meeting minutes:[week5 meeting](#)

In week5, since our team members were all exposed to machine learning for the first time, we got stuck on using machine learning to train the model to recognize the colour of medicine. So we asked our client Andrea about how to use machine learning such as TensorFlow or Pytorch to identify the color of medicine. But in her opinion, it is not a good idea to use machine learning because there are so few data sets. But we still intended to use machine learning, so we decided to keep self-learning to fix the problem.

4.5.1.5. Week6 client meeting

Link to week6 meeting minutes:[week6 meeting](#)

In week6, we completed our machine learning model but the accuracy was about 77%, which is not acceptable. So by the recommendation from clients, we changed to use traditional algorithms. In addition, clients suggested we do the integration work as soon as possible to make sure the final product can export on time. We decided to increase our work efficiency to make the group progress on time.

4.5.1.6. Week7 First Deployment Presentation

Link to zoom recording of First Deployment: [Setdose First Deployment meeting](#) password:0t^Z&AR2

This week, we had the First Deployment Presentation with our clients. The main demo is about three methods we used to recognize medicines, that is using PyTorch to recognize the colour of medicine, using TensorFlow to recognize the colour of medicine, and using OpenCV to recognize the volume of medicine.

4.5.1.7. Week8 client meeting

Link to week8 meeting minutes:[week8 meeting](#)

In week8, we basically completed our register, login, and user profile website and showed the templates to clients in the meeting. However, we still need to add the history function to our user profile web. In addition, clients suggested that it might be possible to share users' information with pharmaceutical companies, we decided to try to implement it. Also, clients asked us to provide an outline of products that can be delivered. We will finish it in week9.

4.5.1.8. Week9 client meeting

Link to our deliverable timeline:[Deliverable product timeline](#)

In week9, our client Rebekah Moles was unable to attend the client meeting on Friday for personal reasons, there was no client meeting in week9. But we still made a summary of the deliverable product timeline. We showed our completed work such as the sign-up page, sign-in page, the home page, recognition result page, user profile page, and history page. We also mentioned that we will continue to connect the functionality with the database, add features that allow customers to edit personal information, deploy the database to the server and design the Android APP.

4.5.1.9. Week10 client meeting

Link to week10 meeting minutes:[week10 meeting](#)

In week10, we reported our group progress and showed our completed development work about web applications to our users. We presented our login website, register website, user profile website, and history function to clients. Clients were very satisfied with our website, but we still noticed that there were some little bugs about our website. For example, our CSS, JS file of login,

and register websites are not working when it connecting by Flask. We need to fix this bug in the following weeks. In addition, we made a little mistake in our presentation, we forgot to open our Redis connection which leads 500 internal errors. We should be more careful and avoid this situation in our Final demo presentation.

4.5.1.10. Week11 client meeting

Link to week11 meeting minutes:[week11 meeting](#)

In week11, we showed clients our newly added administrator functionality for our web and several error pages such as 404 error, 500 error, 403 error, and 400 error. We also fixed the bug that our UI image of the login and register website could not display when connecting by Flask. After that, our clients suggest we make more tests such as API test, user acceptance test, and function test. We decided to handle it in week12.

4.5.1.11. Week12 client meeting

Link to week12 meeting minutes:[week12 meeting](#)

In week12, we showed clients our Android APP that can return the volume and colour of the medicine through the pictures taken by the phone camera. Clients were very satisfied with this APP and suggested we do some tests such as unit tests, API tests, and acceptance tests. We planned to complete these tests before the end of week12.

4.5.1.12. Week13 client meeting

Link to week13 meeting minutes:[week13 meeting](#)

In week13, clients suggested that we could use the opportunity of this client meeting to practice the Final Deployment Presentation. But some of our team members were unable to attend the meeting because of other busy work in week13, so we did not practice the presentation. After the meeting, we confirmed the time of the Final Deployment Presentation with clients and sent them the marking guide of the Final Deployment Presentation.

4.5.2. Screenshot of email

Here is the link to our screenshot of chat history with clients by email and explanation: [Client interaction by email evidence](#)

We basically talked about the time of weekly client meetings, the support we can get, the scope statement, the Confirmation of the time and marking criteria about First Deployment Presentation

and Final Deployment Presentation, the Deliverable Product timeline, and zoom link to each meeting.

4.5.3. Screenshot of zoom chat history

Here is the link to our screenshot of chat history with clients by zoom: [Client interaction by zoom evidence](#)

4.5.4. Scope statement and confirmation

Here is the link to our Scope statement: [scope statement](#)

Here is the link to our Scope statement confirmation: [Scope statement confirmation](#)

In our Scope Statement, it mentioned that the aim of our project is to help nurses and parents ensure that the dose of drugs in syringes is accurate and help reduce the time wasted by manual dose verification and human errors.

4.6. Critique

4.6.1. Advantage

4.6.1.1. Teamwork

4.6.1.1.1. Team execution

If something unexpected happens, for example, consumer AnYan noticed that our login and register website is not working, then we immediately organized a group meeting to handle this emergency, we let someone check the front-end, someone checks the back-end, and someone test after we fixed it. This situation has happened many times, but we can all deal with it quickly and effectively.

4.6.1.1.2. Work enthusiasm

From the weekly group meeting in the first half of the semester to the group meeting two or three times a week in the second half of the semester, each group member keeps full attendance. In addition, before each presentation such first presentation, final presentation, and demo presentation in a client meeting, we will all do a rehearsal of the presentation in Zoom, before ensuring that everyone can speak their corresponding parts fluently, we will continue to rehearse again and again.

4.6.1.2. Group process

4.6.1.2.1. Complete work progress on time

In the second half of the semester, whether XP roles work or development work, each team member can complete the current week's work on time. The manager can submit the weekly status report on time, the Consumer can hold the client meeting on time, and each programmer can complete the assigned tasks on time. This ensures that our group process is always on time.

4.6.1.2.2. The detailed allocation of development work in the second half of the semester

In the second half of the semester, we have accelerated the group progress by reasonably and carefully assigning everyone's web development work. We let one member handle the website front-end, one member creates a database, one member design App, one member handles the website back-end and one member fixes the bug.

4.6.1.3. XP roles

4.6.1.3.1. Frequent client interaction by Customer

Throughout the whole semester, Customer AnYan not only guarantees to hold client meetings on time every week but also frequently communicates with clients by email. Sometimes Customer will ask users about the specific requirements of the product, and sometimes Customer will convey our technical problems to users. When a client or team member can not attend the meeting for some reason, the Customer will report in time to solve the problem. The most important thing is that Customers will show our development work to the clients at the weekly meeting so that clients can know our work stage.

4.6.1.3.2. Reasonable work distribution in web development by Head Programmer

In the second half of the semester, we mainly focus on web development, but there are so many tasks in web development. It involves website front-end, website back-end, and databases, we also need to develop APP. However, the Programmer will arrange a reasonable and necessary amount of work for every group member each week, which leads to our rapid progress in the second half of the semester, allowing us to complete product development in time.

4.6.2. What needs to improve

4.6.2.1. Teamwork

4.6.2.1.1. Lack of test

In our development work in the first half of the semester, we hardly did a test. The reason is that we are new to machine learning and do not clearly know how to test TensorFlow and Pytorch. If possible, we should test the results and the accuracy of the model.

4.6.2.1.2. Uneven teamwork distribution

Also in the first half of the semester, since there was not much development work in the first three weeks, so the teamwork

assigned to each team member is unbalanced. XP roles such as Manager and Consumer will have a larger workload than other roles in the first three weeks. We can let Tester, Doomsayer, and Head programmer take on part of teamwork to avoid this situation.

4.6.2.2. Group process

4.6.2.2.1. Hesitation in development direction

We were very hesitant when deciding how the software should implement the recognition algorithm. We did not know whether to choose colour recognition, volume recognition, or other recognitions, and did not understand the needs of customers, so our group progress was delayed in the first few weeks. Maybe asking our tutor for help or talking to someone with relevant experience will improve the situation.

4.6.2.2.2. Process deliverable in a hurry

Due to uneven teamwork distribution and hesitation in product direction, our progress was very slow at the beginning, which resulted in us completing our demo development almost at the end of the week6, which means that we only have a few days to write our First report, it took us one or two day to rush the report, at last, the First report only got 70%. At that time, we didn't realize the seriousness of the problem. If we convened a group meeting earlier to explain to each group member that our time is not enough and make corresponding countermeasures, perhaps we can avoid this situation from happening.

4.6.2.3. XP roles

4.6.2.3.1. Confusion about early work by Head Programmer, Tester, and Doomsayer

It's the first time for us to participate in such a large and complex project, so everyone has no experience, and in the first few weeks we did not have much development work. Head Programmer, Tester, and Doomsayer were unfamiliar with what they needed to do. Next time maybe we can be more familiar with the part of the responsibility of Manager, Consumer, and Tracker.

4.6.2.3.2. Initial XP roles job confusion

Since all the team members were not familiar with the XP role's responsibilities at the beginning, there were problems in the division of tasks for each person. For example, the tracker did the work that should be the customer, and the customer did the work that should be the manager. We should learn about XP roles before or ask our tutor for help.

Reference

- [1]. Eyler, R., & Mueller, B. (2021). The Hidden Pandemic of Misdosing. Hospitals.
- [2]. Sharabiani, A., Nutescu, E. A., Galanter, W. L., & Darabi, H. (2018). A new approach towards minimizing the risk of misdosing warfarin initiation doses. Computational and mathematical methods in medicine, 2018.
- [3]. Australian Commission of Safety and Quality in healthcare.(2020).Quality Use of Medicines and Medicines Safety
- [4]. Children's Hospital of The King's Daughters. (2017). subcutaneous injection at Home. Retrieved from <https://www.chkd.org/patients-and-families/health-library/way-to-grow/subcutaneous-injection-at-home/>
- [5]. Aronson, J. K. (2009). Medication errors: what they are, how they happen, and how to avoid them. QJM: An International Journal of Medicine, 102(8), 513-521.

Codebase

- [The source code of the web application](#)
- [The source code of the Android application](#)

The links above are the source code for the Web application in our project and the source code for the lightweight version of Android. Because the environments and configuration requirements of the two applications are so different, we did not combine them into a single version. We thought it would be clearer to present them separately.