Macau University of Science and Technology

Faculty of Innovation Engineering

DIAD
Dangerous Item Auto Detection System

Luo Yemao 2009853G-I011-0041

Luo Wei 2009853G-I011-0070

Lin Longjun 2009853F-I011-0014

2302 SE252 D1

May 2023

Contents

1 Requirements Document

1.1 Problem

For the current Security inspection system, We have collected following problems:

1) Due to the large volume of X-ray images that need to be analyzed every day and the limited number of security personnel available to detect hazardous items, there is a significant risk of errors occurring.

2) Employing a substantial number of security personnel to address the identified challenge would require a significant allocation of financial resources. This would involve conducting various recruitment drives, providing training, and administration of the personnel. Additionally, compensation for the security personnel, which is critical in retaining them, would necessitate a sustained budgetary allocation.

1.2 Background Information

1) Public transport is a location where large numbers of people gather, making it vulnerable to significant losses in the event of accidents. The implementation of a Dangerous Item Detection System could help detect hazardous items such as explosives, thereby ensuring public safety. By doing so, incidents that could cause harm to public security can be prevented.

2) The high volume of public transportation poses a significant challenge for security personnel to manually screen every individual's package using X-ray machines. Implementation of a Dangerous Item Detection System could greatly alleviate this burden as it automatically detects hazardous items. This would help reduce the workload on security personnel.

1.2.1 Evolutionary Model

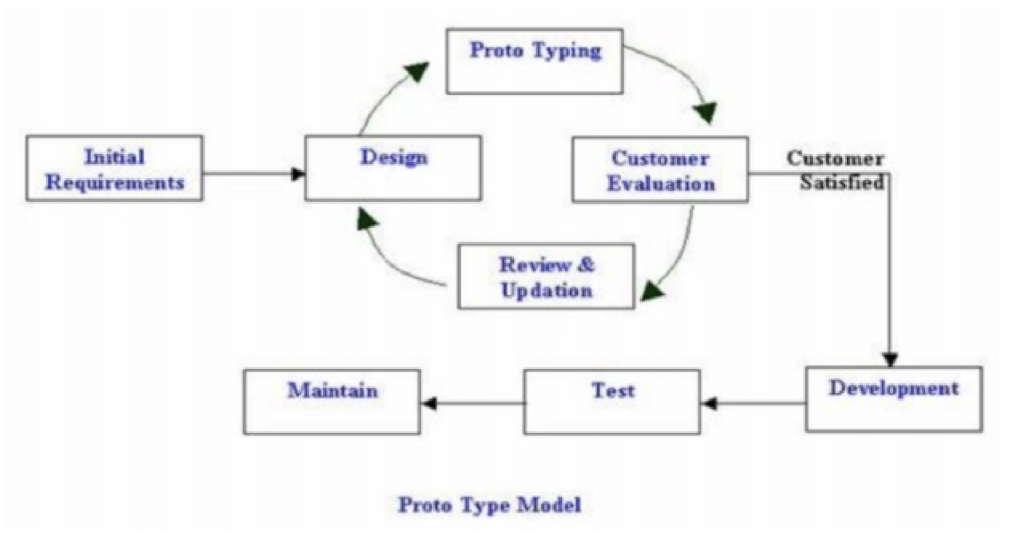


Figure 1: Evolutionary Model

Obviously, we cannot define all the requirements of our DIAD in advance. That's the reason why we use the evolutionary model. For implementation, We need to communicate core requirements to users during the development process so that they are able to provide effective feedback when we see the core requirements implemented to support the final design and implementation of the system. When the core system is put into operation after development, users try it out first, complete their work, and put forward new requirements to refine the system and enhance the system's capability. Software developers implement an iterative process of development based on user feedback.

1.3 Environment and System Models

Our system is designed to be adaptable to different environments, thereby ensuring its efficient operation and availability across various platforms. This versatility allows it to run seamlessly on different operating systems, which are the common platforms in use today, including Windows, Linux, and Apple systems. Therefore, users with different technological preferences can use our system without the need for operating system change or purchase of specific hardware and software.

1.4 Use Case Diagram

Use Case Diagram:

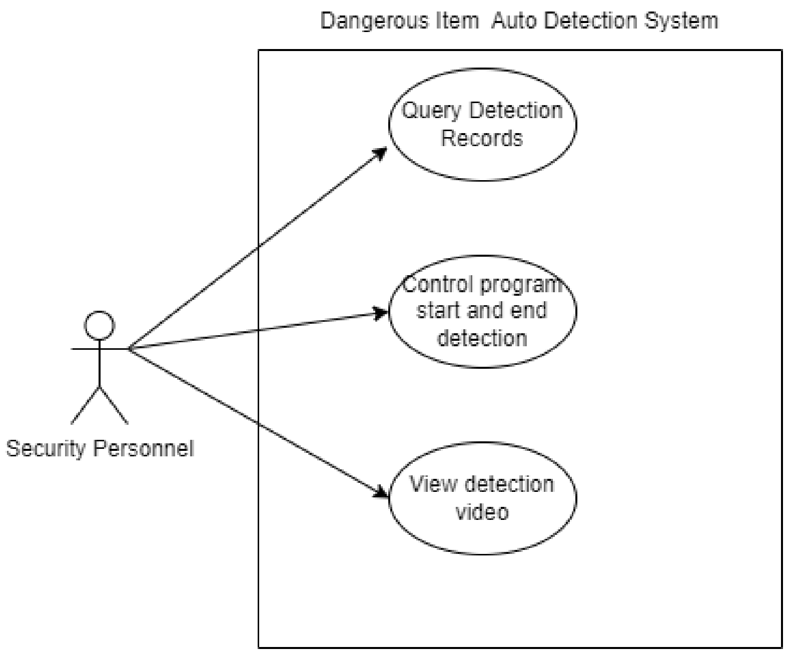

Dangerous Item Auto Detection System
Query Detection Records
Control program start and end detection
View detection video
Security Personnel

Figure 2: Use Case Diagram

1.5 Functional Requirements

1) Security personnel can view the currently playing detection video.
2) Security personnel can click the start detection button to start running the automatic detection program.
3) Security personnel can click the stop detection button to terminate the automatic detection program.
4) Security personnel can check past testing records.
5) When dangerous goods are detected, the system can provide strong reminders to security personnel.

1.6 Non-Functional Requirements

1.6.1 Detection Accuracy

The DIAD system serves as an automated detection software responsible for identifying hazardous materials present in public transportation. Despite its efficiency, any missed detection of such dangerous goods would constitute a significant risk to public safety, resulting in potentially severe social harm. This makes it imperative to deploy a software solution that exhibits the highest level of detection accuracy.

1.6.2 Reliability

As a security inspection machine is required to operate for extended periods without interruption or failure, it must exhibit considerable stability. This implies that the system should maintain a constant level of CPU usage and avoid any sudden spikes or dips that can compromise its performance. In addition, the system must not flash unexpectedly or crash during operation. These critical requirements constitute the fundamental criteria for the seamless, stable operation of any security inspection machine over an extended period.

1.6.3 Compatibility

As a result of the varying computer configurations present in security machines, it is imperative that our software solution can operate efficiently and smoothly across multiple operating systems, including Windows, Linux, and Mac OS. The compatibility of our program with different platforms ensures that it can function seamlessly on different hardware and software configurations, thereby eliminating any technical barriers that may hinder the effective operation of the security machine.

1.6.4 Operability

Considering that the target user group for the DIAD system is national security personnel, the software must guarantee ease of use, thereby avoiding excessive training costs. Achieving this requires the inclusion of clear and well-structured instructions on the user page, which can be easily understood by users with varying levels of technical proficiency. The primary objective of this approach is to enable users to quickly become familiar with the system's interface and functionality, thus minimizing the time and resources required for training.

1.6.5 Portability

As an existing security machine typically comes with its own security system, it is paramount that the DIAD system integrates seamlessly with this original system. Therefore, it must be designed in a manner that ensures compatibility with all corresponding pre-installed security systems simultaneously. This approach guarantees a reliable and efficient system that leverages the state-of-the-art capabilities of existing security machines.

1.6.6 Security

Recognizing the critical impact of a security inspection system on ensuring the safety of the entire public transportation system, it is crucial that the DIAD system be designed with robust security features to prevent any unauthorized access. The system must robust encryption mechanisms to safeguard against potential threats to public safety. With effective security measures in place, the DIAD system can help prevent security incidents that could otherwise endanger society.

1.6.7 Performance

Given the high number of people using public transportation and the corresponding large volume of items that must be detected and screened within a limited timeframe, the detection system must be highly responsive. To address this demand, the system must be capable of rapidly detecting and processing a large number of items within a short period. These requirements underpin the crucial importance of developing detection software that can operate at high speed to maximize efficiency and mitigate any potential risks.

2 Design Document

2.1 Purpose

The purpose of Dangerous Item Auto Detection (DIAD) is to provide an automated solution for detecting dangerous items in X-ray images, primarily aimed at enhancing the security screening process in public transportation settings. DIAD aims to expedite and streamline the identification of hazardous objects within luggage or personal belongings, ensuring the safety and security of passengers and public transport systems.

Specifically, the main purposes of DIAD are as follows:

1) Automated Detection: DIAD employs advanced computer vision and machine learning algorithms to automatically analyze X-ray images and identify potential dangerous items. By leveraging pattern recognition and object classification techniques, DIAD can efficiently detect items such as firearms, explosives, knives, and other prohibited or hazardous objects.

2) Real-time Screening: DIAD is designed to operate in real-time, enabling quick and accurate detection of dangerous items during the security screening process. By reducing manual inspection and relying on automated analysis, DIAD improves the efficiency and throughput of security checkpoints, ensuring minimal disruptions to the flow of passengers.

3) Increased Security: By providing a robust and reliable detection mechanism, DIAD enhances security measures in public transportation. It helps security personnel identify potential threats promptly, enabling them to take appropriate actions to mitigate risks and maintain a safe environment for passengers and staff.

4) User-friendly Interface: DIAD features a user-friendly interface for security personnel to interact with the system effectively. It provides intuitive controls, clear visualizations of X-ray images, and informative alerts or notifications for detected dangerous items. This facilitates seamless integration of DIAD into existing security protocols and ensures ease of use for operators.

In summary, the purpose of Dangerous Item Auto Detection (DIAD) is to provide an automated solution for detecting dangerous items in X-ray images, facilitating efficient and accurate security screening in public transportation. By leveraging advanced technologies, DIAD enhances security measures, improves the speed of inspections, and contributes to maintaining a safe and secure travel experience for passengers.

2.2 General Priorities

2.2.1 System Stability

In a public transportation dangerous item detection scenario, maintaining system stability is paramount due to the need for continuous operation. The DIAD system must ensure high operational uptime, and be resilient against various possible disruptions, such as hardware failures or power outages. Also, the system should have built-in redundancies and the ability to perform self-checks to quickly identify and recover from any potential malfunctions.

2.2.2 System Performance

Given the high volume of X-ray images that need to be analyzed continuously, the DIAD system is designed to be highly performant and efficient. The system is optimized to handle a large number of screen detection processes simultaneously, enabling multiple security personnel to use the system without causing any degradation in performance.

The main computational task of the DIAD system involves the analysis of X-ray images using a pre-trained machine learning model. These images, grabbed from a designated region on the screen, are analyzed in real-time to detect potential threats. It's important to note that the system is optimized to process the images directly without the need for storage, thus considerably enhancing the system's speed and performance.

Our code is designed to be memory-efficient as well. It processes each X-ray image independently and does not need to store them, thus minimizing memory usage. This design decision ensures that the system can run smoothly even when dealing with a high volume of X-ray images.

Finally, potential threats identified in each image are rapidly reported to the security personnel. This efficient detection and reporting mechanism ensure that any potential hazards can be swiftly addressed. This mechanism ensures rapid reporting of potential threats to the security personnel for further action.

2.2.3 Diversity of Function

The DIAD system not only focuses on dangerous item detection, but also provides a monitoring component for tracking CPU, Memory, and Disk usage. This feature would enable the administrative staff to ensure the smooth running of the system and take necessary actions in case of high resource usage. Further, the system will also provide functionality to analyze past detection records, providing insights into to analyze past detection records, providing insights into the frequency and type of hazardous items detected, thereby aiding strategic decision-making.

2.2.4 Aesthetics

While the system needs to be functional, we also understand the importance of an intuitive and pleasant user interface. The interface would be designed to present the necessary information clearly without overwhelming the user. Care will be taken to balance aesthetics with performance, ensuring that the design doesn't compromise the speed of detection and reporting.

2.3 Outline of The Design

2.3.1 Project architecture

The DIAD system will be built following the service-oriented architecture pattern. We plan to split the system into two main components - the detection module and the reporting module. This design would effectively decouple the system business logic and manage the growing complexity, making it easier for agile development and maintenance.

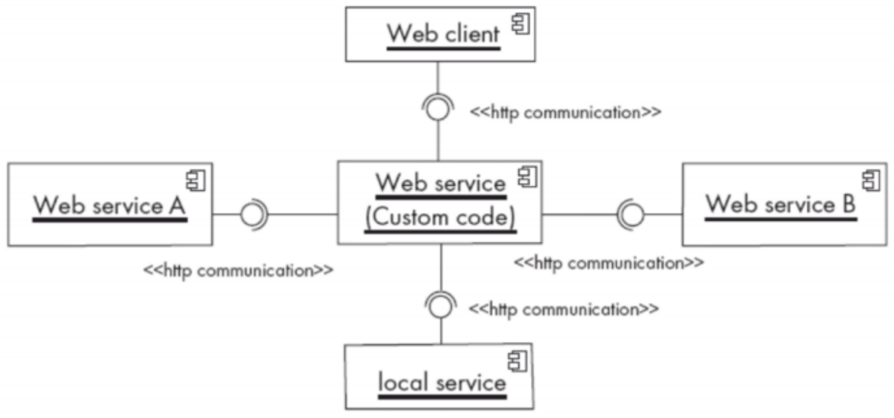


Figure 3: Service-Oriented Architecture

2.3.2 Principle of Design

1) Divide and conquer: This approach breaks the system down into manageable modules, each dealing with a specific part of the functionality.

2) Reduce coupling where possible: The system is designed in such a way that the interdependence between modules is minimal. This reduces the ripple effect of changes and errors, thereby enhancing the system's stability and maintainability.

3) Keep the level of abstraction as high as possible: High-level abstraction is maintained throughout the system design to hide the complexity of the underlying implementation. This helps in making the system easier to manage, understand, and modify.

4) Reuse existing designs and code where possible: Existing, proven designs and code are used wherever possible to accelerate the development process and reduce the risk of errors. Open-source libraries and frameworks have been employed where applicable.

5) Design for flexibility: The system is designed to adapt to evolving requirements. This includes using modular design principles, and ensuring that components and model can be added, modified, or removed with minimal impact on other parts of the system.

6) Design for Portability: The DIAD system is designed to be portable, capable of operating across a variety of hardware platforms and operating systems. This is achieved through adherence to standard protocols and avoidance of platform-specific features.

7) Design for Testability: DIAD is designed with testability in mind. It includes features that facilitate testing, such as modularity.

8) Anticipate obsolescence: In a rapidly evolving technological landscape, the system design accounts for future changes and upgrades, ensuring that DIAD remains effective and relevant over time. This is achieved by choosing technologies and designs that are forward-compatible, as well as providing easy pathways for upgrades and expansions.

2.4 Major Design Issues

2.4.1 Back End

1) The Dangerous Item Detection System (DIAD) operates continuously, analyzing a set frequency of X-ray images to identify hazardous items. This constant load on the system intensifies the performance demand. Designing a system that can efficiently handle this steady high load without degradation in performance or accuracy is a major challenge.

2) Ensuring that the system correctly and consistently identifies dangerous items is critical. There are many strategies for achieving this, but we need to strike a balance between detection accuracy and system performance. A highly accurate detection system that uses complex algorithms can place a heavy load on the system and degrade performance. On the other hand, a system that prioritizes performance over accuracy might miss some dangerous items, potentially compromising public safety.

2.4.2 Front End

1) The user interface needs to be designed with consideration for system performance and usability. Complex visuals and extensive detail can slow down the system, which is undesirable given the need for rapid response in security situations. Therefore, it's a challenge to balance the aesthetics of the user interface with the need to maintain efficient, reliable system performance.

2) With regards to the usability of the system, it is important to consider the ease of use for the security personnel who will be operating it. A design that is too complicated or unintuitive could lead to operator errors or slower response times. Therefore, designing a user interface that is intuitive and easy to use, while still providing all necessary information and functionality, is a significant design challenge.

2.5 Class Diagram

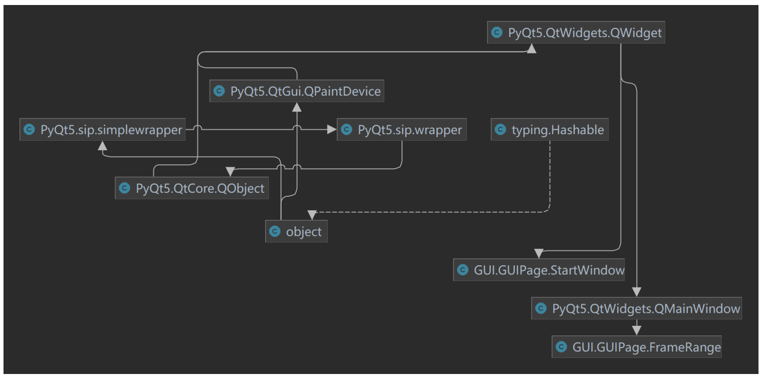


Figure 4: Class Diagram 1

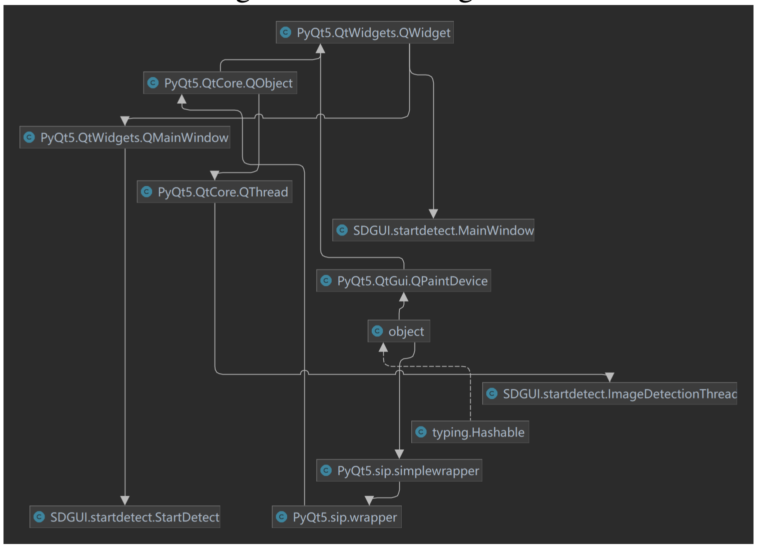


Figure 5: Class Diagram 2

2.6 Sequence Diagram

Sequence diagram is a UML interaction diagram. It shows the dynamic collaboration among multiple objects by describing the time sequence of sending messages between objects. It can represent the sequence of use case behavior. When a use case behavior is executed, each message in it corresponds to a class operation or trigger event in the state machine that causes the transition. We draw the process of security personnel using our software to detect X-ray hazardous items.

First, the user clicks "Start" on the initial page and selects the screenshot area. The program then passes the screenshot area to the screenshot detection page and captures the screenshot. At this point, the program sends the screenshot to the backend inference process. Finally, the inference process returns the annotated detection image of hazardous items along with their categories, which is displayed on the screenshot detection page.

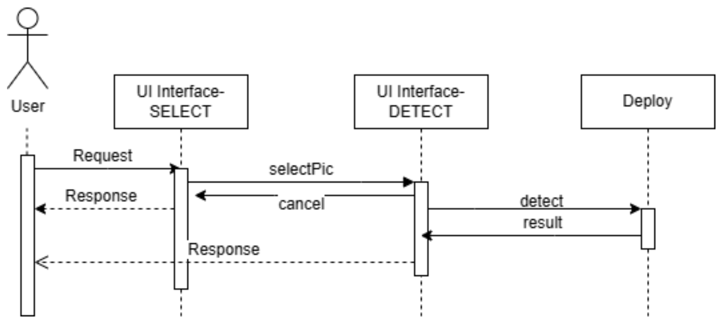


Figure 6: User Log In

3 Development Document

3.1 Development method

3.1.1 Product development method

The purpose of our DIAD (Dangerous Item Auto Detection) project is to provide security personnel, specifically those working in public transportation security, with a reliable and efficient tool for detecting dangerous items in X-ray images. By automating the detection process, DIAD aims to enhance the speed and accuracy of identifying potential threats, thereby improving the overall effectiveness of security checks. The system's primary objective is to ensure the safety and well-being of passengers by minimizing the risk of dangerous items being transported onto public transportation vehicles.

3.1.2 Pros

- Large project is divided into manageable amounts.
- System developments are coded and tested during each sprint review.
- Works well for fast-moving development projects.
- The individual effort of each team member is visible at any time.

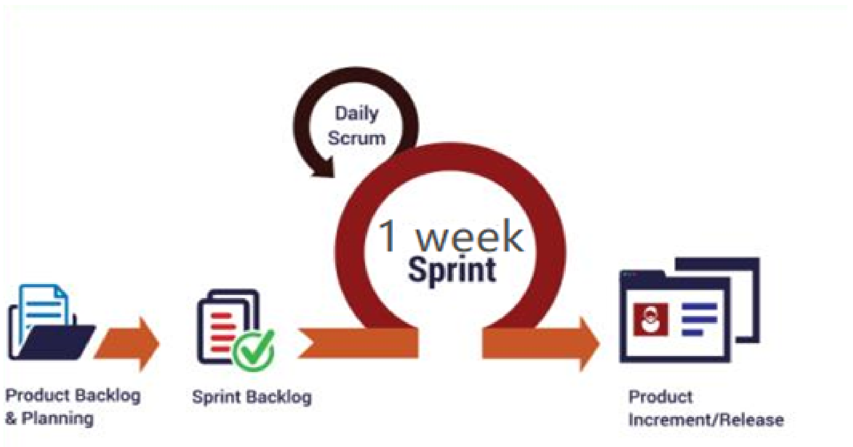


Figure 7: Sprint planning and arrangement

3.2 Team member roles and duties

3.2.1 Team member

- ***Product Manager: Luo Yemao***

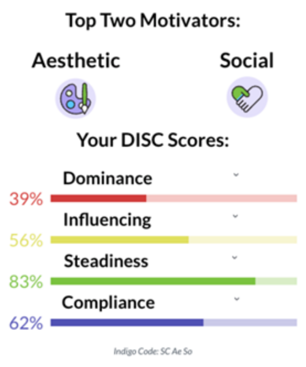
  1) Close partners with the business and the team to ensure everyone understands the work items in the product backlog.

2) Give the team clear guidance on which features to deliver next.
3) Decide when to ship the product with a preference towards more frequent delivery.
4) Ensure that the development team delivers the most value to the business.

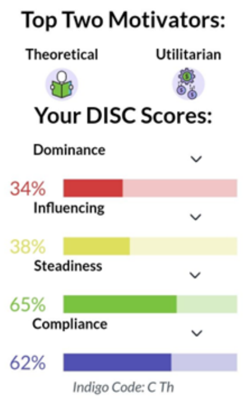
- ***Development team: Luo Yemao, Luo Wei, Lin Longjun***

1) The TEAM is responsible for converting the product backlog into potentially shipable increments of functionality at each iteration.
2) The TEAM is self-managing, has actual autonomy, and must be culturally compatible, based on motivating people's initiative and avoiding external interference.
3) The TEAM has the full right to decide how to convert requirements into product functions, such as whether to do design, what algorithm to use, how to do defect prevention, etc.
4) Neither the PO nor the SM has the right to direct the TEAM to realize the requirements, but the TEAM must promise to deliver The paid function is what the PO expects.

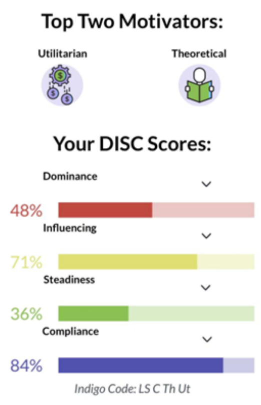
3.2.2 Evaluation of Our Team Members' Personality



(a) Luo Yemao



(b) Lin Longjun



(c) Luo wei

Figure 8: Main Personality From DISC

Luo Yemao, our project manager, exhibits a steady and patient S-type personality. He demonstrates a strong focus on the needs of others and maintains a calm and reliable demeanor. These traits contribute to creating a harmonious team dynamic, where everyone feels valued and heard. Luo Yemao's patience and empathy are assets when it comes to addressing conflicts and maintaining a positive working environment.

Lin Longjun, possesses an S-type personality. Her patient and stable nature proves to be valuable in her role, particularly when working on tasks related to training models. Given the complex nature of these tasks, which demand meticulous attention to detail, Lin Longjun's patience and stability become advantageous traits. Her ability to remain focused and committed ensures that the training process is thorough and accurate, leading to high-quality outcomes.

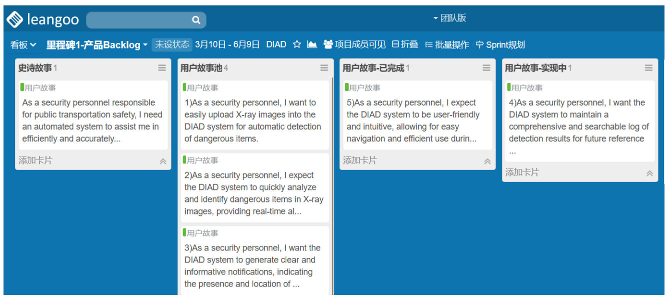
Luo wei, a team member responsible for UI design and other plugins, possesses a C-type personality. She demonstrates an analytical and detail-oriented approach, which greatly benefits tasks requiring precision and accuracy. Luowei's meticulous work style ensures that every aspect is thoroughly addressed, resulting in a high-quality end product. She follows a clear plan of action and adheres to deadlines, making her a valuable asset to our project.

These distinct personality traits complement the team's skill set and contribute to the overall success of the project. By leveraging our strengths, we can effectively handle complex tasks, maintain attention to detail, and deliver a high-quality outcome within the defined timelines.

3.3 Development Process

3.3.1 Epic Story

As a security personnel responsible for public transportation safety, I need an automated system to assist me in efficiently and accurately identifying potential threats in X-ray images during security checks. The system needs to analyze and recognize dangerous items in X-ray images, providing timely alerts and improving public transportation safety. By implementing this system, we aim to enhance the overall security measures and ensure a safer travel experience for passengers.



史诗故事 1

用户故事
As a security personnel responsible for public transportation safety, I need an automated system to assist me in efficiently and accurately...

用户故事池 4

用户故事
1)As a security personnel, I want to easily upload X-ray images into the DIAD system for automatic detection of dangerous items.

用户故事
2)As a security personnel, I expect the DIAD system to quickly analyze and identify dangerous items in X-ray images, providing real-time al...

用户故事
3)As a security personnel, I want the DIAD system to generate clear and informative notifications, indicating the presence and location of ...

用户故事-已完成 1

用户故事
5)As a security personnel, I expect the DIAD system to be user-friendly and intuitive, allowing for easy navigation and efficient use durin...

用户故事-实现中 1

用户故事
4)As a security personnel, I want the DIAD system to maintain a comprehensive and searchable log of detection results for future reference ...

Figure 9: Epic Story

3.3.2 User Story

In order to meet the needs of the user, we divided the epic story into many user stories to facilitate the arrangement of sprints. So as to better complete the development task as soon as possible.

1) As a security personnel, I want to easily upload X-ray images into the DIAD system for automatic detection of dangerous items.

2) As a security personnel, I expect the DIAD system to quickly analyze and identify dangerous items in X-ray images, providing real-time alerts.

3) As a security personnel, I want the DIAD system to generate clear and informative notifications, indicating the presence and location of potential dangerous items.

4) As a security personnel, I want the DIAD system to maintain a comprehensive and searchable log of detection results for future reference and analysis.

5) As a security personnel, I expect the DIAD system to be user-friendly and intuitive, allowing for easy navigation and efficient use during security checks.

6) As a security personnel, I want the DIAD system to be reliable and accurate, minimizing false positives and false negatives in the detection process.

7) As a security personnel, I need the DIAD system to be compatible with existing X-ray scanning equipment and seamlessly integrate into our security workflow.

8) As a security personnel, I expect the DIAD system to undergo regular updates and improvements to keep up with evolving security threats and ensure optimal performance.

3.3.3 Product Backlog

We have decided two parts of project in order to satisfy these three epic stories. First epic story fits in with our first part which is PROJECT_ADMIN which is mainly focused on the management platform. The second and third epic stories fit in with our second part which is PROJECT_COES which is mainly focused on the teacher and student interface.

1) Training and fine-tuning of deep learning models using annotated datasets

2) Development of user-friendly graphical user interface (GUI) for ease of use by security personnel

3) Integration of X-ray image capture and processing functionality

4) Implementation of real-time object detection algorithms for efficient item recognition

5) Integration of alert generation and notification system for potential threats

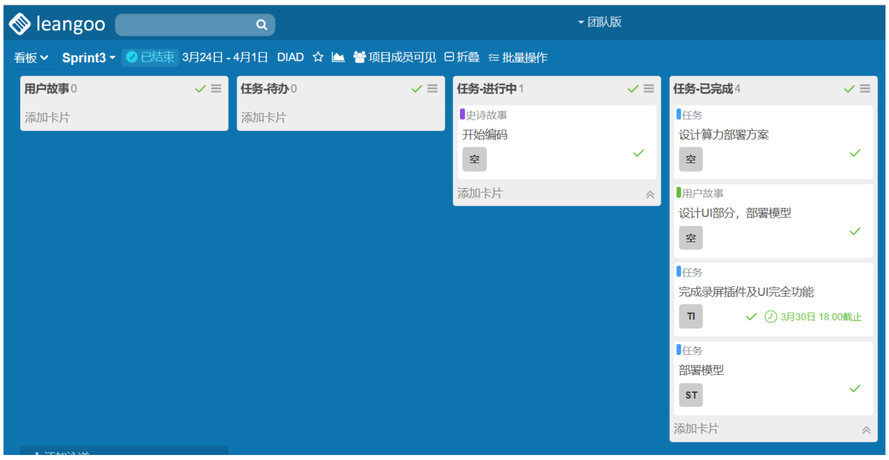
6) Extensive testing and debugging to ensure system reliability and accuracy

7) Performance optimization and resource utilization improvement

8) Implementation of system logging and monitoring for security and audit requirements

9) Post-deployment evaluation and resolution of any issues or enhancements

10) Ongoing maintenance and updates to meet evolving security requirements


leangoo

团队版

看板 Sprint3 已结束 3月24日 - 4月1日 DIAD 项目成员可见 折叠 批量操作

用户故事 0

添加卡片

任务-待办 0

添加卡片

任务-进行中 1

史诗故事

开始编码

空

添加卡片

任务-已完成 4

任务

设计算力部署方案

空

用户故事

设计UI部分，部署模型

空

任务

完成录屏插件及UI完全功能

TI

3月30日 18:00截止

任务

部署模型

ST

添加卡片

Figure 10: Sprint’s Kanban (example of sprint 3)

3.3.4 Sprints & Burn Down Charts

- **Sprint1 – Establish Overall Project Goals and Design**

Meeting Name	**Meeting Minutes**	**Sprint Tasks**
Sprint Planning Meeting	Discuss and determine the goals and plans for project goals and design.	Define project goals and initial design concepts.
Daily Stand-up Meeting	Report on the progress in establishing project goals and design	Discuss the project goals and potential design options.
Sprint Review Meeting	Review the work completed during the sprint and discuss plans for the next sprint.	Review the established project goals and design.
Sprint Retrospective Meeting	Review and reflect on the work completed during the sprint and make suggestions for improvement.	Discuss the effectiveness of the goal setting and design process and propose improvements for the next sprint.

- **Sprint2 – Initiate Model Training and Design User Interface Sketches**

Meeting Name	Meeting Minutes	Sprint Tasks
Sprint Planning Meeting	Discuss and determine the goals and plans for model training and UI design.	Initiate the model training process and design UI sketches.
Daily Stand-up Meeting	Report on the progress in model training and UI design.	Discuss the progress of model training and UI design.
Sprint Review Meeting	Review the work completed during the sprint and discuss plans for the next sprint.	Review the initiation of model training and UI design.
Sprint Retrospective Meeting	Review and reflect on the work completed during the sprint and make suggestions for improvement.	Discuss the effectiveness of the model training and UI design process and propose improvements for the next sprint.

- **Sprint3 – Deploy Model and Begin Developing User Interface**

Meeting Name	Meeting Minutes	Sprint Tasks
Sprint Planning Meeting	Discuss and determine the goals and plans for model deployment and UI development.	Deploy the trained model and begin developing the user interface.
Daily Stand-up Meeting	Report on the progress in model deployment and UI development.	Discuss the progress of model deployment and UI development.
Sprint Review Meeting	Review the work completed during the sprint and discuss plans for the next sprint.	Review the deployment of model and the beginning of UI development.
Sprint Retrospective Meeting	Review and reflect on the work completed during the sprint and make suggestions for improvement.	Discuss the effectiveness of the model deployment and UI development process and propose improvements for the next sprint.

- **Sprint4 – Establish Overall Project Goals and Design**

Meeting Name	**Meeting Minutes**	**Sprint Tasks**
Sprint Planning Meeting	Discuss and determine the goals and plans for designing and executing tests.	Design and execute tests to ensure the accuracy and performance of the program.
Daily Stand-up Meeting	Report on the progress in test design and execution.	Discuss the progress of test design and execution.
Sprint Review Meeting	Review the work completed during the sprint and discuss plans for the next sprint.	Review the designed tests and their execution.
Sprint Retrospective Meeting	Review and reflect on the work completed during the sprint and make suggestions for improvement.	Discuss the effectiveness of the test design and execution process and propose improvements for the next sprint.

- **Burn Down Chart**

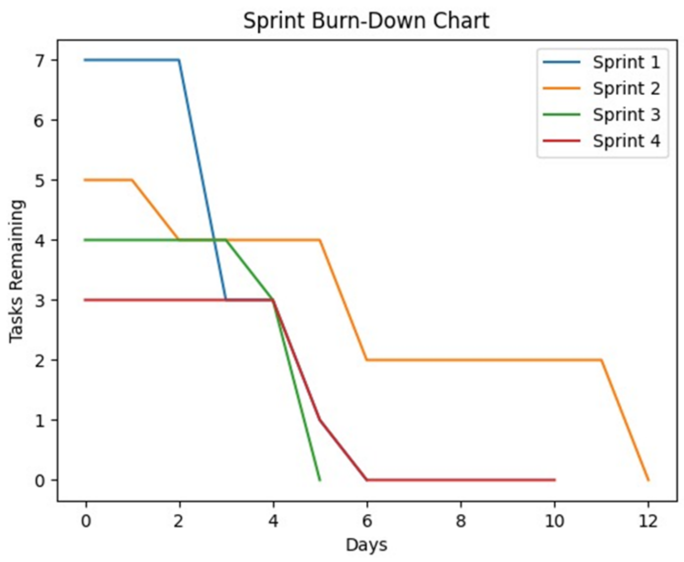


Figure 11: Burn Down Chart

4 Testing Document

Software testing is a critical process that involves executing software to identify and correct errors and improve reliability, correctness, and safety. The ultimate aim is to demonstrate that the software meets design requirements, prevents defects, and finds any that may exist. However, deciding on the most effective software testing approach can be challenging in the software life cycle. After confirming user requirements and discussing with software QA engineers, we opted for agile testing.

There were several reasons why we chose agile testing, including the fact that the project involved parallel development and testing, which meant that the overall project time was faster. Additionally, work tasks were clearly divided, and work efficiency was high. Agile testing also allowed us to identify and solve bugs in a short time, minimizing their impact on related modules. Effective communication with developers throughout the testing process reduced the number of bugs detected.

We began testing on March 20th and continued until the final presentation on May 11th. The agile testing section was divided into three sub-headings: black box testing, white box testing, and automatic testing. Black box testing was conducted using four methods to obtain the final functional test cases. These included the boundary value analysis method, causality diagram method, and decision table method. In black-box testing, the focus was on those three parts. In white box static testing, we reviewed code, performed code walk-throughs, and used code-checking tools. In white box dynamic testing, three methods were employed to design the testing case, including the condition-covered method, path-covered method, and flow control table. We also used a spectrum-based fault location tool-Afluent to locate possible code lines that may have caused unit test failures.

4.1 Black-box testing

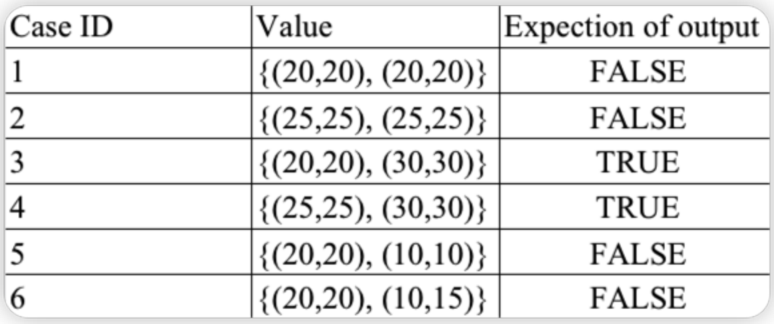
4.1.1 Boundary Value Analysis

Many program errors occur at the boundaries of the input or output range. Therefore, setting test cases for various boundary conditions can help to identify many program defects. Boundary value analysis is a method used to identify such errors. It is different from equivalence class division, where a representative value is randomly chosen from a certain equivalence class. In boundary value analysis, each boundary of the equivalence class is used as a test condition. This approach not only considers input conditions but also considers test conditions generated by the output space.

Based on the principles of boundary value analysis, we tested the function of points-scoring. By testing the function at the boundaries of the input and output range, we were able to identify potential errors and ensure that the function worked correctly within the specified range. This technique is an effective way to identify and correct errors in software development, as it helps to ensure that the software meets design requirements and is reliable and effective. Therefore, we use this method to test our function when the user is selecting an area on the screen to take a screenshot. By doing so, we hope we can reduce the possibility of an error occurring in this first stage of using our software.

Area input
Test requirements: input an area's start point and end point n to judge if it is a valid area.
Analysis: Determine the effective area and invalid area.
Critical point: {(0,0), (0,0)}
To ensure a valid scenario, the start point and end point must meet specific requirements. Firstly, the location of the start point and end point cannot be the same. If they are the same, it is an invalid scenario. Secondly, the endpoint cannot be higher than the start point. If the endpoint is higher than the start point, it is still a valid scenario.

By following these requirements, we can ensure that the scenario is valid and meets the necessary conditions for the software to function correctly. This is an essential consideration when developing software, as it ensures that the software operates as expected and is reliable and effective.



Case ID	Value	Expection of output
1	{(20,20), (20,20)}	FALSE
2	{(25,25), (25,25)}	FALSE
3	{(20,20), (30,30)}	TRUE
4	{(25,25), (30,30)}	TRUE
5	{(20,20), (10,10)}	FALSE
6	{(20,20), (10,15)}	FALSE

Figure 12: Boundary Value Test case

4.2 White-box Testing

4.2.1 Static White Box Testing

4.2.1.1 Code Review

Code review is a process of improving the quality of code by rechecking it by others in the software life cycle. Code review involves team members checking each other's code systematically and consciously to ensure that it meets the required quality standards.

Conducting code reviews in the retrospective meeting during each sprint is an excellent practice. During the code review process, the code developer should introduce the code they have written, and the leader should provide feedback and comments on Github or in person, pointing out any mistakes or areas that could be improved. Once the code has passed the review process and been approved by the leader, it can be merged into the master branch.

Code review is an essential part of the software development process as it helps to identify and correct errors, ensure code consistency, improve code maintainability, and increase the overall quality of the code. By conducting regular code reviews, software development teams can ensure that their code meets the necessary quality standards and is reliable and effective.

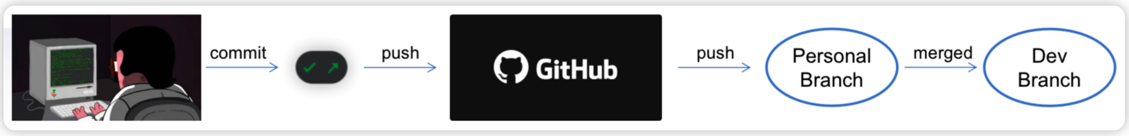


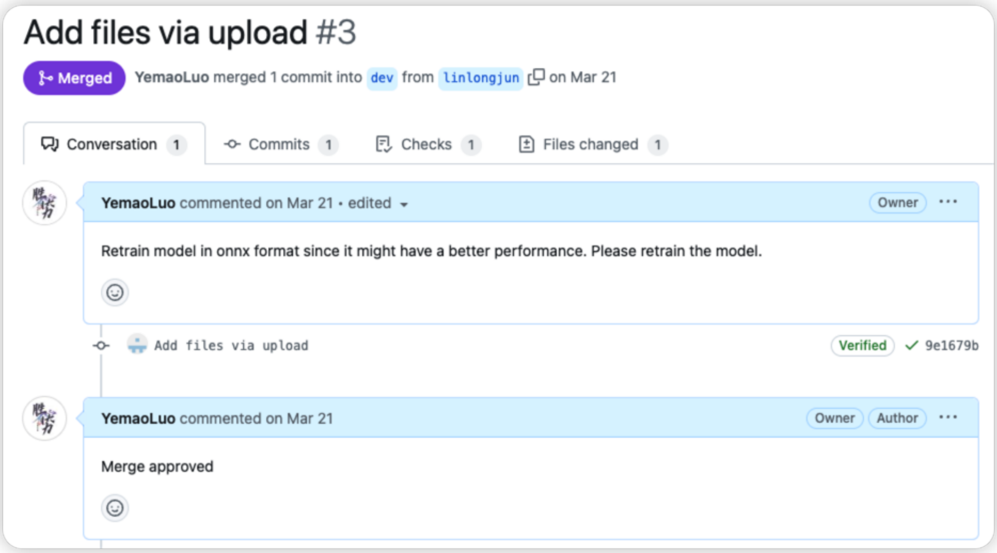
Figure 13: Process of Developing with Code Review

4.2.1.2 Code Walk-through

It is good practice to hold code walkthroughs irregularly, as this ensures that the code is checked thoroughly at different stages of development.

During the walkthroughs, the code developer should introduce the code they have written, and other developers in the group should inspect it. The inspection should focus on identifying errors, including issues with data definition, variable field effects, and parameter conveyance methods. By identifying and correcting these errors, the team can ensure that the code is reliable, effective, and meets the necessary quality standards.

Overall, code walkthroughs are an important part of the software development process, and by conducting them regularly, software development teams can ensure that their code is of high quality and meets the necessary standards for reliability and effectiveness.


Add files via upload #3

Merged YemaoLuo merged 1 commit into dev from linlongjun on Mar 21

Conversation 1 Commits 1 Checks 1 Files changed 1

YemaoLuo commented on Mar 21 • edited Owner

Retrain model in onnx format since it might have a better performance. Please retrain the model.

Add files via upload Verified ✓ 9e1679b

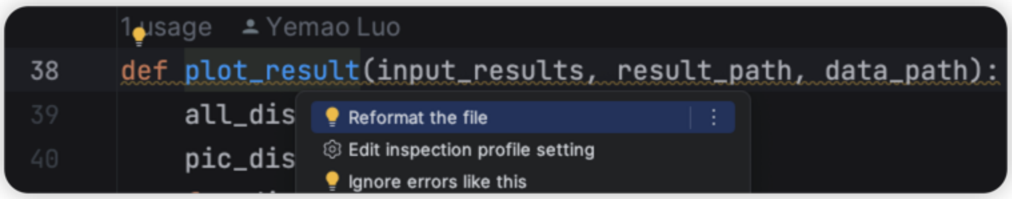
YemaoLuo commented on Mar 21 Owner Author

Merge approved

Figure 14: Walk-through

4.2.1.3 Static Testing Tools

We use a built-in inspection plugin to automatically check for grammar errors in our code, and we follow the coding guideline to standardize our coding style. By using the inspection plugin, we can identify and correct common mistakes and errors in our code, ensuring that it meets the necessary quality standards. Additionally, standardizing our coding style according to the coding guideline helps to ensure that our code is consistent, maintainable, and easily understood by other developers. This enables us to write high-quality code that is reliable and effective in the long term.



```
1 usage  Yemao Luo
def plot_result(input_results, result_path, data_path):
    all_dis
    pic_dis
```

Figure 15: Static Code checking Tool

4.2.2 Dynamic White Box Testing

Static white box testing is a process that involves a thorough review of software design, architecture, and code in a systematic manner, without actually executing the software. This process is also known as structural analysis, and it is used to identify software defects that may be difficult to find or isolate using dynamic black box testing.

One of the primary advantages of static white box testing is that it enables software defects to be identified as early as possible. This is important because defects found early in the development process are typically easier and less expensive to fix than those found later. Additionally, by focusing on software design review at the beginning of the development process, the testing team can provide valuable insights and feedback to the development team, which can help to improve the overall quality of the software.

Another important benefit of static white box testing is that it can provide ideas and guidance for black box testers when designing and applying test cases during software testing. By identifying potential defects and weaknesses in the software design and architecture, the testing team can design more effective test cases that cover a wider range of scenarios and edge cases.

In summary, static white box testing is a valuable process that can help to identify software defects early in the development process and provide insights and guidance for black box testing. By incorporating static white box testing into the software development life cycle, software development teams can improve the quality and reliability of their software.

4.2.2.1 Control Flow Diagram

A control flow diagram is an abstract representation of a procedure or program, and it is also an abstract data structure used within compilers to represent all possible paths traversed during the execution of a program. This diagram represents the potential flow direction of all basic block execution in a process in form of a diagram, and it can also reflect the real-time execution of a process.

To gain a better understanding of the internal logic of a program, flow control tables can be designed. For example, let us consider the functions of login and making comments. As the lines of code for each function can be quite far apart, a description of the function rather than the code line number can be used to create the flow control table. This allows for a clear and concise representation of the program's logic, which can aid in the debugging and optimization processes.

Overall, control flow diagrams and flow control tables are valuable tools for software developers as they allow for a clear representation of a program's logic and potential execution paths. By using these tools, software development teams can improve the quality and reliability of their software and ensure that it meets the necessary standards for effectiveness and functionality.

Below is the flow chart of the whole process of using the software. The ring complexity of the flow chart is 5.

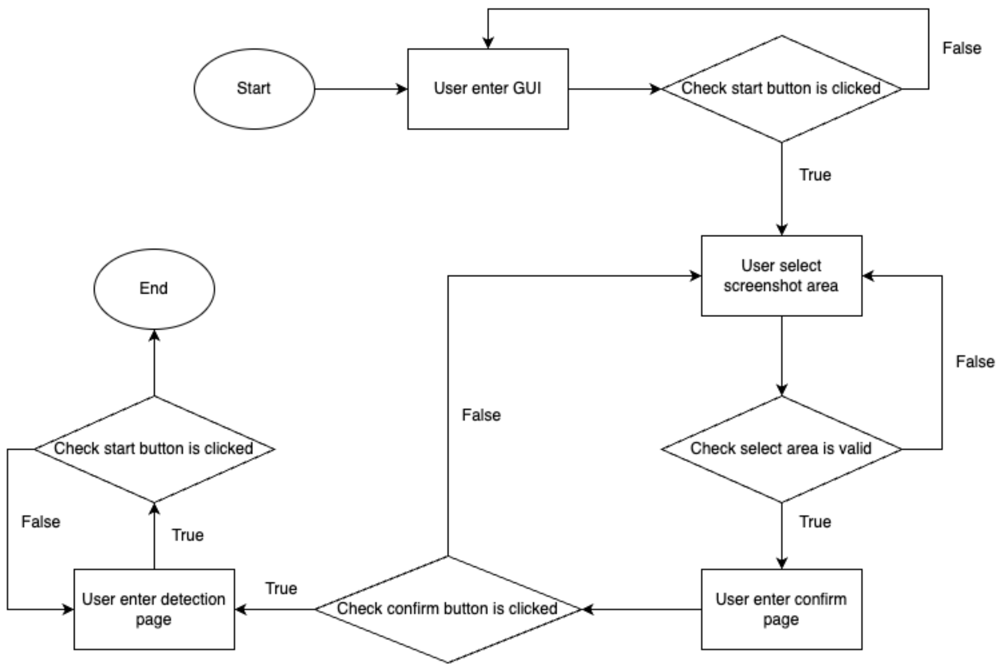

Start
User enter GUI
Check start button is clicked
False
True
User select screenshot area
False
Check select area is valid
True
User enter confirm page
Check confirm button is clicked
False
True
User enter detection page
True
Check start button is clicked
False
End

Figure 16: Flow Chart

4.2.2.2 Independent Path Coverage

Independent path testing is a testing methodology that is based on the program's control flow diagram. By analyzing the loop complexity of the control structure and deriving the basic executable path collection, this methodology is used to design the corresponding test case method. The test cases that are designed using this methodology should ensure that the statements of the program cover 100% and the conditions cover 100% in the test.

In order to conclude the dependent paths, we can use the flow control table. The dependent paths are those that are not covered by the independent test cases, and they must be tested separately to ensure that the program is thoroughly tested. By using both independent path testing and dependent path testing, software development teams can ensure that their programs are reliable, effective, and meet the necessary quality standards. All independent paths are given as follows.
Path1: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10
Path2: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 8 -> 10
Path3: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10
Path4: 1 -> 2 -> 3 -> 4 -> 5 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10
Path5: 1 -> 2 -> 3 -> 2 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10

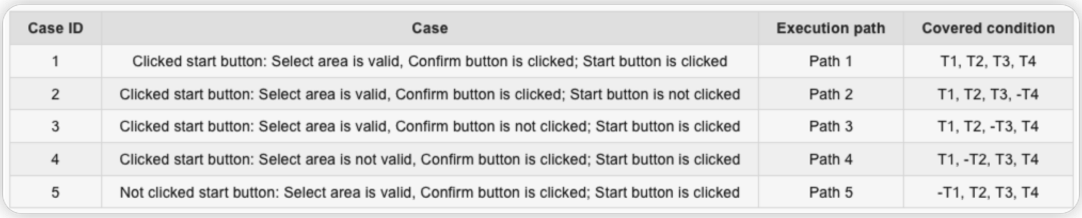
4.2.2.3 Condition Coverage Method

Although decision and conditional coverage meet the criteria for decision and conditional coverage, they do not guarantee coverage of all condition combinations within the decisions of the tested code. As a result, conditional combination coverage is necessary to cover all condition combinations in the tested code.

In the software-using process, we have 4 conditions:
Clicked start button, true is T1,false is -T1
Select area is valid, true is T2, false is -T2
Confirm button is clicked, true is T3, false is -T3
Start button is clicked, true is T4, false is -T4

4.2.2.4 Testing Cases

For the analysis of the above conditional coverage and path coverage, we design the following test cases.



Case ID	Case	Execution path	Covered condition
1	Clicked start button: Select area is valid, Confirm button is clicked; Start button is clicked	Path 1	T1, T2, T3, T4
2	Clicked start button: Select area is valid, Confirm button is clicked; Start button is not clicked	Path 2	T1, T2, T3, -T4
3	Clicked start button: Select area is valid, Confirm button is not clicked; Start button is clicked	Path 3	T1, T2, -T3, T4
4	Clicked start button: Select area is not valid, Confirm button is clicked; Start button is clicked	Path 4	T1, -T2, T3, T4
5	Not clicked start button: Select area is valid, Confirm button is clicked; Start button is clicked	Path 5	-T1, T2, T3, T4

Figure 17: Test Cases

4.3 Automatic Testing

Automated testing has become an essential part of software testing in recent years, as it offers a more efficient and reliable way to execute test cases. By converting test procedures into machine-executable programs, automated testing eliminates the possibility of human error and reduces the time and effort required for testing.

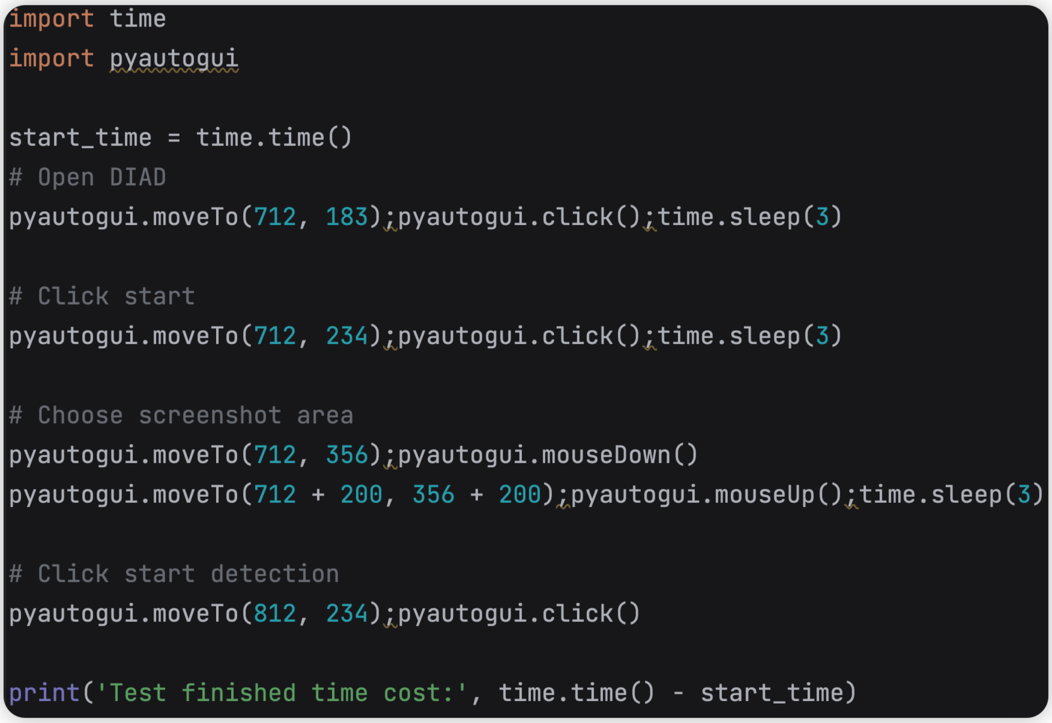
The process of automated testing involves generating test data and scripts, which are then used to simulate test behaviors based on predefined test cases. This can be done through a variety of techniques, such as record-and-playback, keyword-driven, or data-driven testing, depending on the specific requirements of the testing scenario.

Automated testing also enables testers to run tests more frequently and consistently, which is especially important in agile development environments, where new code is frequently added and tested. Additionally, automated testing can help identify defects earlier in the development process, when they are easier and less expensive to fix.

Overall, automated testing offers a number of benefits over manual testing, including improved efficiency, reliability, and speed. By automating repetitive and time-consuming tasks, testers can focus on more complex and critical areas of software testing, ultimately improving the quality and reliability of the software being tested.

4.3.1 Visual Bug Testing

We use pyautogui to execute customized commands in order to find the visual bugs.



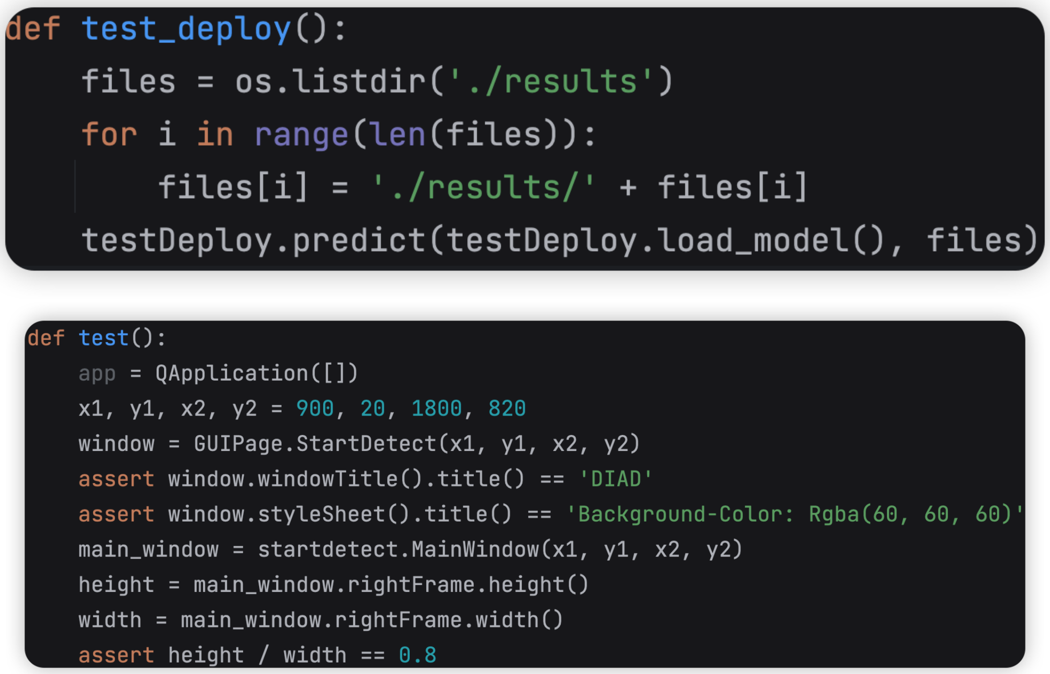
```python
import time
import pyautogui

start_time = time.time()
# Open DIAD
pyautogui.moveTo(712, 183);pyautogui.click();time.sleep(3)

# Click start
pyautogui.moveTo(712, 234);pyautogui.click();time.sleep(3)

# Choose screenshot area
pyautogui.moveTo(712, 356);pyautogui.mouseDown()
pyautogui.moveTo(712 + 200, 356 + 200);pyautogui.mouseUp();time.sleep(3)

# Click start detection
pyautogui.moveTo(812, 234);pyautogui.click()

print('Test finished time cost:', time.time() - start_time)
```

Figure 18: Customized command lines

4.3.2 Unit Testing

Unit testing is one of the most rigorous means of software testing and is the testing method that is closest to the underlying implementation of the code. It ensures the quality of local code at the lowest cost in the early stages of software development. Since unit tests are executed in an automated manner, they can bring high benefits in the scenario of a large number of regression tests.

Implementing unit tests can also help development engineers improve the design and implementation of code. Unit tests provide examples of how functions are used in the code, as they are expressed by calling functions with various combinations of input parameters. These calling methods constitute the instructions for using functions. We used Pytest to perform unit tests on some functions in our software. Among these unit tests, one test failed.

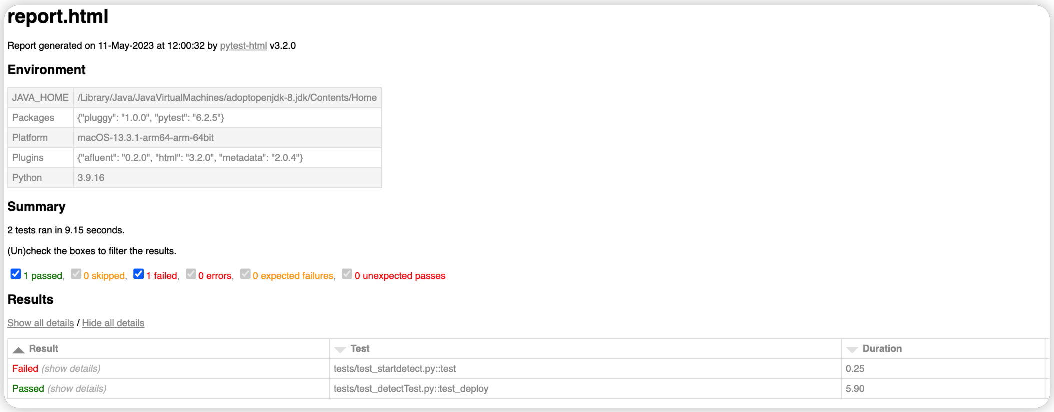


```
def test_deploy():
    files = os.listdir('./results')
    for i in range(len(files)):
        files[i] = './results/' + files[i]
    testDeploy.predict(testDeploy.load_model(), files)
```

```
def test():
    app = QApplication([])
    x1, y1, x2, y2 = 900, 20, 1800, 820
    window = GUIPage.StartDetect(x1, y1, x2, y2)
    assert window.windowTitle().title() == 'DIAD'
    assert window.styleSheet().title() == 'Background-Color: Rgba(60, 60, 60)'
    main_window = startdetect.MainWindow(x1, y1, x2, y2)
    height = main_window.rightFrame.height()
    width = main_window.rightFrame.width()
    assert height / width == 0.8
```

Figure 19: Unit Testing Code

After executing the two unit tests described above, we obtained the following results. One of the test cases passed successfully, indicating that the associated code performed as expected. However, the other test failed, indicating that there was an issue with the code that needs to be addressed. Further investigation is necessary to identify the root cause of the failure and to make the necessary modifications to the code to ensure that it performs correctly.



report.html

Report generated on 11-May-2023 at 12:00:32 by pytest-html v3.2.0

Environment

JAVA_HOME	/Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home
Packages	{"pluggy": "1.0.0", "pytest": "6.2.5"}
Platform	macOS-13.3.1-arm64-arm-64bit
Plugins	{"afluent": "0.2.0", "html": "3.2.0", "metadata": "2.0.4"}
Python	3.9.16

Summary

2 tests ran in 9.15 seconds.

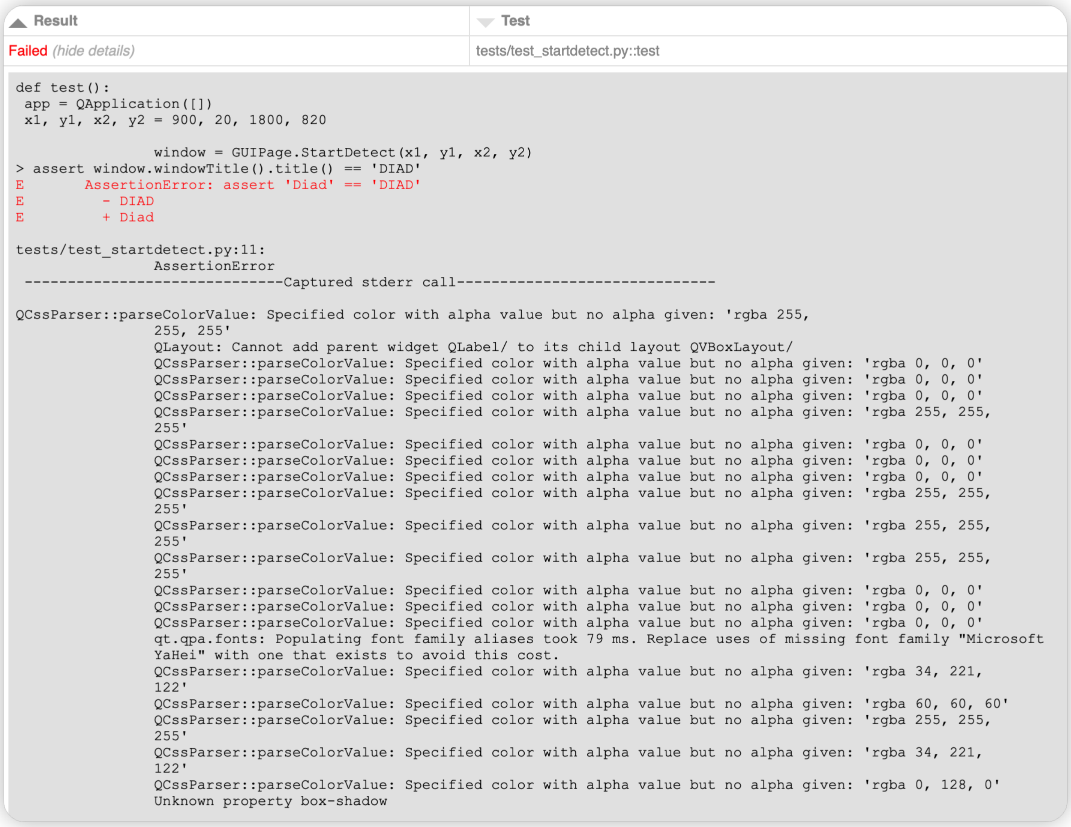
(Un)check the boxes to filter the results.

✓ 1 passed, ✓ 0 skipped, ✓ 1 failed, ✓ 0 errors, ✓ 0 expected failures, ✓ 0 unexpected passes

Results

Show all details / Hide all details

Result	Test	Duration
Failed (show details)	tests/test_startdetect.py::test	0.25
Passed (show details)	tests/test_detectTest.py::test_deploy	5.90



Result	Test
Failed (hide details)	tests/test_startdetect.py::test

```
def test():
 app = QApplication([])
 x1, y1, x2, y2 = 900, 20, 1800, 820

                window = GUIPage.StartDetect(x1, y1, x2, y2)
> assert window.windowTitle().title() == 'DIAD'
E       AssertionError: assert 'Diad' == 'DIAD'
E         - DIAD
E         + Diad

tests/test_startdetect.py:11:
                AssertionError
 -----------------------------Captured stderr call-----------------------------

QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 255,
                255, 255'
                QLayout: Cannot add parent widget QLabel/ to its child layout QVBoxLayout/
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 0, 0, 0'
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 0, 0, 0'
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 0, 0, 0'
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 255, 255,
                255'
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 0, 0, 0'
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 0, 0, 0'
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 0, 0, 0'
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 255, 255,
                255'
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 255, 255,
                255'
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 255, 255,
                255'
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 0, 0, 0'
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 0, 0, 0'
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 0, 0, 0'
                qt.qpa.fonts: Populating font family aliases took 79 ms. Replace uses of missing font family "Microsoft
                YaHei" with one that exists to avoid this cost.
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 34, 221,
                122'
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 60, 60, 60'
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 255, 255,
                255'
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 34, 221,
                122'
                QCssParser::parseColorValue: Specified color with alpha value but no alpha given: 'rgba 0, 128, 0'
                Unknown property box-shadow
```

Figure 20: Unit Testing Result

4.3.3 Spectrum-based Fault Localization

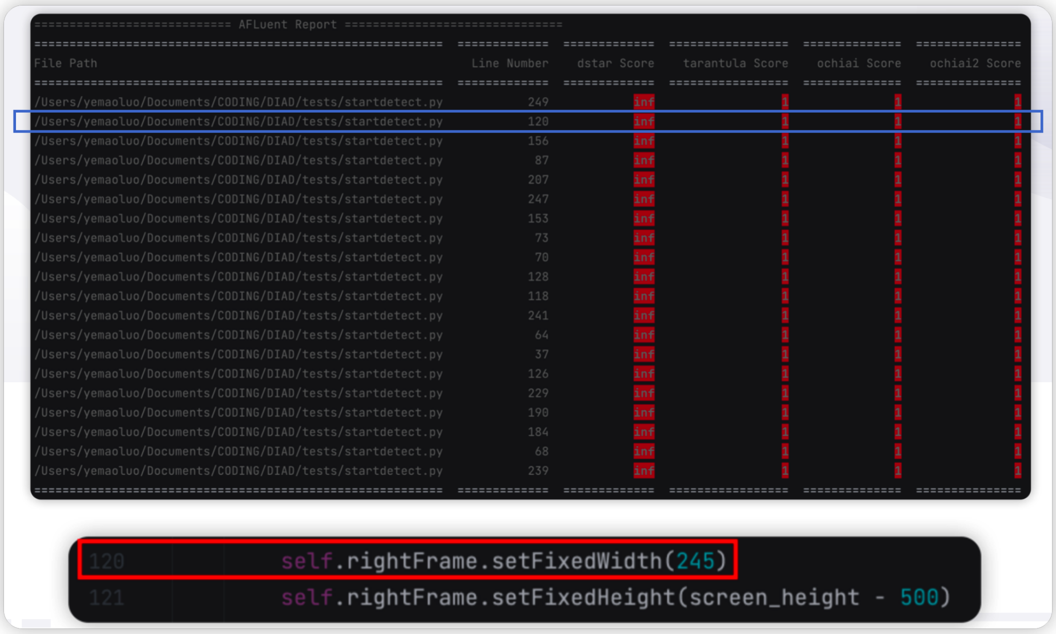
For the defect localization tool, we have chosen AFluent, which is an open-source spectrum-based fault localization analysis tool on GitHub.

https://github.com/AFLuent/AFLuent

AFLuent is an automated fault localization tool built as a Pytest plugin. It's triggered when one or more test case fails causing it to generate a ranking of suspicious statements where the cause of the fault could possibly be. Statements are ranked in descending order based on a score calculated through information from code coverage and using one of four supported equations (Tarantula, Ochiai, Ochiai2, and DStar).

AFluent serves as a Pytest plugin that automatically starts when a unit test fails. In the previous chapter, when the unit test failed, AFluent ran and printed out all possible code lines that caused this error. We were able to successfully identify the fault in the second line of the code, which was indicated to be error code line.

Figure 21: AFluent Logo



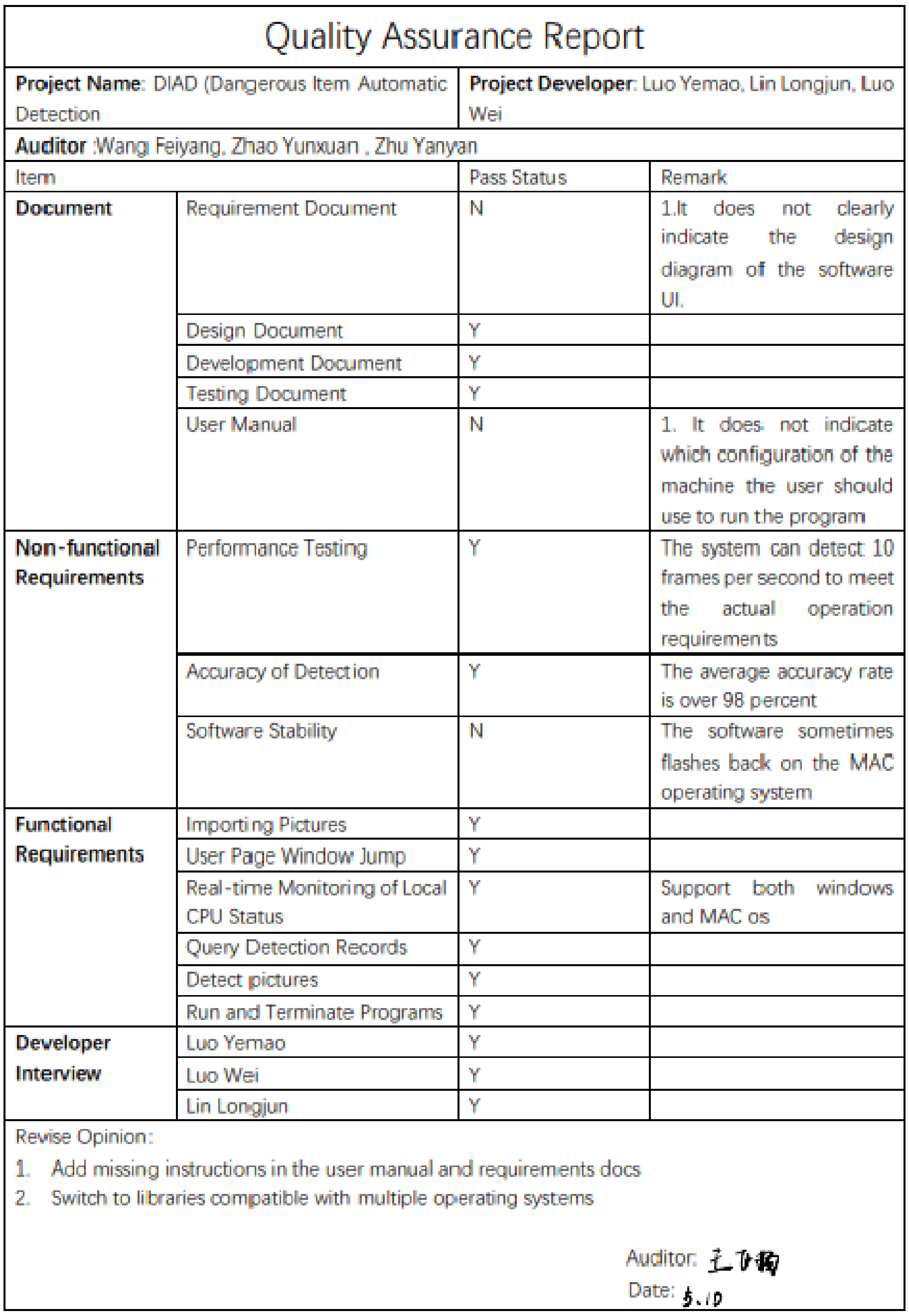
```
=========================== AFLuent Report ===============================
============================================================ ============ ============ ================= ============== ===============
File Path                                                     Line Number  dstar Score  tarantula Score  ochiai Score   ochiai2 Score
============================================================ ============ ============ ================= ============== ===============
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py           249          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py           120          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py           156          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py            87          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py           207          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py           247          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py           153          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py            73          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py            70          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py           128          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py           118          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py           241          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py            64          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py            37          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py           126          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py           229          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py           190          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py           184          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py            68          inf                 1              1               1
/Users/yemaoluo/Documents/CODING/DIAD/tests/startdetect.py           239          inf                 1              1               1
============================================================ ============ ============ ================= ============== ===============
```

```
120        self.rightFrame.setFixedWidth(245)
121        self.rightFrame.setFixedHeight(screen_height - 500)
```

Figure 22: Report and Error Code Line

4.4 Quality Assurance

To enhance the quality of our software, we have enlisted Wang Feiyang to perform quality assurance testing on our software. They are responsible for ensuring the quality of our documents, functional and non-functional requirements, as well as the work of our developers. The resulting quality assurance report in Figure 23 includes the pass/fail status of each item, along with any remarks indicating reasons for failure. Additionally, improvement suggestions for the project are provided.



Quality Assurance Report

Project Name: DIAD (Dangerous Item Automatic Detection

Project Developer: Luo Yemao, Lin Longjun, Luo Wei

Auditor :Wang Feiyang, Zhao Yunxuan , Zhu Yanyan

Item		Pass Status	Remark
Document	Requirement Document	N	1.It does not clearly indicate the design diagram of the software UI.
	Design Document	Y	
	Development Document	Y	
	Testing Document	Y	
	User Manual	N	1. It does not indicate which configuration of the machine the user should use to run the program
Non-functional Requirements	Performance Testing	Y	The system can detect 10 frames per second to meet the actual operation requirements
	Accuracy of Detection	Y	The average accuracy rate is over 98 percent
	Software Stability	N	The software sometimes flashes back on the MAC operating system
Functional Requirements	Importing Pictures	Y	
	User Page Window Jump	Y	
	Real-time Monitoring of Local CPU Status	Y	Support both windows and MAC os
	Query Detection Records	Y	
	Detect pictures	Y	
	Run and Terminate Programs	Y	
Developer Interview	Luo Yemao	Y	
	Luo Wei	Y	
	Lin Longjun	Y	

Revise Opinion:

1. Add missing instructions in the user manual and requirements docs
2. Switch to libraries compatible with multiple operating systems

Auditor: [illegible]
Date: 5.10

Figure 23: Quality Assurance

5 Management and Maintenance

Software management involves all aspects of software development, and its direct targets include people, money, and things. Simply put, people refer to software developers, money refers to project funds, and things refer to software projects. As a software manager, you should stand on high ground and overlook the entire project. With the premise of the awareness of overlooking the overall situation and the use of appropriate management techniques, the project will be easier to carry out.

5.1 Software Management

5.1.1 Process Matric

Our project management decision proceeds from four aspects, which are Scope, Time, Cost, and Quality. Scope, time, and cost together make the Quality triangle, popularly known as the project management triangle.

1) Scope, that is, the scope of work refers to all the work that needs to be done in order to achieve the final goal. By scope, we can indicate "what needs to be done to achieve the project goals", or "what work can be done and can be closed". This is very important if, without a defined scope of work, projects may be difficult or never complete. Therefore, we must strictly control changes in the scope of work. Once there is a change, it is likely to affect the expected goals and final delivery results of the project.

2) Time, which is the time required to complete the task goal, can be used to help us quantify the number of tasks and the expected completion time of the task. Time is also the most important indicator in our task plan. Time specifies the start and end time of a specific task.

3) Cost refers to all money consumed during the software development process, including a series of expenses such as personnel salaries and equipment expenses. Of course, these expenses are not unlimited. We need an overall budget. When the project is completed, the overall expenses should be within the budget within.

4) Quality refers to the degree to which the output meets expectations. Generally speaking, the degree to which we judge whether a project meets expectations is the quality of the project. We can also put forward requirements on the quality of the project according to our expectations so that the completion of the project is more in line with expectations.

The goal of our team is to try to find a balance among time, cost, and scope, so as to achieve the most perfect quality expectation under a reasonable balance of the three. So as to achieve our expectation of adopting agile development "sooner and better"!

5.1.2 Project Size-Oriented Metrics

5.1.2.1 Line of Code

During the initial phases of the project, an estimation of the number of lines of code is made. The project is divided into various categories, including model building, software development, testing, and deployment, which are further subdivided as presented in the table 1 below. The estimated number of code lines per section is determined by calculating the average of the number of code lines found in analogous parts of the project that have been researched on Github.

Part	Detail	Estimated Number of Code Lines(KLOC)
Detection Model	Training Model	13
Body of Software	User Interface	0.15
	Screen Record	0.2
Testing	Unit Testing	0.25
	Performance Testing	0.2
Deployment	Local Deployment	0.3
		Total Lines of Code=14.1

Table 1: Estimation of Line Codes

5.1.2.2 Errors in Project

During the initial phases of the project, an estimation of the number of errors is made. The project is also divided into various categories, including model building, software development, testing, and deployment, which are further subdivided as presented in the table 2 below. The estimated number of errors per section is determined by calculating the average of the number of errors found in analogous parts of the project that have been researched on Github.

Part	Detail	Estimated Number of Errors
Detection Model	Training Model	10
Body of Software	User Interface	3
	Screen Record	2
Testing	Unit Testing	0
	Performance Testing	0
Deployment	Local Deployment	2
		Total Errors=17

Table 2: Estimation of Line Codes

5.1.2.3 Estimation

We conducted a study to determine the average daily code output of our team members during the initial three weeks, and our findings indicate that the daily code production was 0.2k. By incorporating the average daily code production of 0.2k into the formula for estimating task duration, we arrive at an estimated completion time of 70.5 days.

$$\begin{aligned} Estimated\ Work &= \frac{LOC}{Estimated\ Productivity} \\ &= \frac{14.1K}{0.2Kper\ day} \\ &= 70.5days \end{aligned}$$

We conducted a comprehensive search for salary information about software developers, algorithm engineers, and project managers on several recruitment websites, including 58.com, boss, and so on. The data obtained from each website were averaged to derive our estimated daily salary rates, which were found to be 170 for software developers, 190 for algorithm engineers, and 230 for project managers. By incorporating the average daily wage into the formula for estimating the cost, we arrive at an estimated cost of 41,595 yuan.

$$\begin{aligned} Estimated\ Cost &= Estimated\ Work \times Daily\ Wage \\ &= 70.5 \times (170 + 190 + 230) \\ &= 41,595 \end{aligned}$$

By substituting the total amount of code calculated before, the total cost, and the total errors, we can estimate the cost of the line and the error of the line.

$$\begin{aligned} Estimated\ Cost\ of\ Line &= \frac{Estimated\ Cost}{LOC} \\ &= \frac{41,595}{14.1K} \\ &= 2950(cost/KLOC) \end{aligned}$$

$$\begin{aligned} Estimated\ Error\ of\ Line &= \frac{Estimated\ Error}{LOC} \\ &= \frac{17}{14.1K} \\ &= 1.21(error/KLOC) \end{aligned}$$

5.1.3 Functional Point Analyse

The estimation of functional points for the project is determined by utilizing the number of input and output types.

Type of Files	Number of Files
Interior Logical File(ILF)	5
Exterior Interface File(EIF)	0
Exterior Input(EI)	1
Exterior Output(EO)	2
Exterior Query(EQ)	1

Table 3: Number of Functional Files

$$\begin{aligned} Functional\ Point(FP) &= 10ILF + 7EIF + 4EI + 5EO + 4EQ \\ &= 10 \times 5 + 7 \times 0 + 4 \times 1 + 5 \times 2 + 4 \times 1 \\ &= 68 \end{aligned}$$

In addition to the functional points, we have taken into account several technical complexity factors and made the necessary adjustments. The effects of the technical complexity impact factors are presented in the table 4 below.

Type of Factors	Technical Complexity
Data communication	4
Distributed Data Process	3
Performance	4
System Equipment Requirement	5
Transaction Rate	4
Online Data Input	2
User Efficiency	3
Online Update	1
Complex Process	3
Reusability	3
Install Ability	3
Operability	3
Workplace Diversity	5
Changeability	1

Table 4: Technological Complexity

$$\begin{aligned} Functional\ Point(FP) &= [(4+3+4+5+4+2+3+1+3+3+3+3+5+1)*0.01+0.65] \times 14 \\ &= [0.44+0.65] \times 68 \\ &= 74.12 \end{aligned}$$

Given that the estimation is conducted during the intermediate and advanced phases of the project, an adjustment factor of 1 is employed to derive the final estimate of functional points.

$$\begin{aligned} Functional\ Point(FP) &= FP \times CF \\ &= 74.12 \times 1.0 \\ &= 74.12 \end{aligned}$$

After adjustment, the Functional points become 74.12.

5.1.4 Software Cost Estimation

5.1.4.1 Cocomo Model

The intermediate COCOMO model was employed to estimate the software workload, utilizing the degree of magnitude and a collection of cost drivers, which encompass objective evaluations of product, hardware, personnel, and project attributes. This extension comprises four categories of cost drivers, each of which comprises several minor attributes. Cost driver can be seen in the following table 5.

Cost Driver	Sub attribute	Assessment
Product Attribute	Reliability	1.4
	Library Size	1.16
	Complexity of Product	1.15
Hardware Attribute	Performance Constraints	1.3
	Memory Constraints	1.06
	Virtual Machine Stability	1.3
	Requirement of Response Time	1.15
Member Attribute	Analytical Capabilities	1.00
	Software Engineering Skills	1.00
	Experience in Application	0.70
	Experience in Virtual Machine	1.00
	Experience in Programming Language	0.95
Project Attribute	Software Tools	0.91
	Software Engineering Methods	0.91
	Development time requirements	1.00

Table 5: Cost Drivers and Assessment

To obtain the workload adjustment factor, we substitute the available factor values obtained from the evaluation into the formula.

$$\begin{aligned} EAF &= 1.4 \times 1.16 \times 1.15 \times 1.3 \times 1.06 \times 1.3 \times 1.15 \times 1 \times 1 \times 0.7 \times 1 \times 0.95 \times 0.91 \times 0.91 \times 1 \\ &= 2.11874507 \end{aligned}$$

Because DIAD is a Semi-Detached Type project, we adjusted the parameters to a=3.0, b=1.12, c=2.5, and d=0.35.We can use these parameters to estimate the total workload, development time, and number of developers.

$$\begin{aligned} E\ (Effort) &= 3 \times (14.1\ KLOC)^{1.12 \times 2.12} \\ &= 123.19 \end{aligned}$$

$$\begin{aligned} D\ (DevelopmentTime) &= 2.5 \times (123.19)^{0.35} \\ &= 13.48\ days \end{aligned}$$

$$\begin{aligned} P\ (People) &= \frac{E}{D} \\ &= \frac{123.19}{13.48} \\ &= 9.14\ persons \end{aligned}$$

5.2 Management Model

5.2.1 Egoless Team Organization

Egoless programming is a state of mind in which programmers are supposed to separate themselves from their product. In this team organization goals are set and decisions are made by group consensus. Here group, ‘leadership’ rotates based on tasks to be performed and the differing abilities of members.

5.2.2 CMMI

The full name of CMMI is Capability Maturity Model Integration. CMMI is the latest version of the CMM model. Early CMMI (CMMI-SE/SW/IPPD), SEI began to promote and try in some countries and regions. With the promotion of applications and the development of the model itself, deduction has become a comprehensive model that is widely used. The full name of CMMI is Capability Maturity Model Integration. There are 5 levels of CMMI certification, CMMI level 1, completion level; CMMI level 2, management level; CMMI level 3, definition level; CMMI level 4, quantitative management level; CMMI level 5, optimization level. We define our team as CMMI level 5 for the following reasons. In project management, we have fully utilized Leangoo for digital management and progress monitoring to achieve management accuracy, so we have already met CMMI level 4. Secondly, we have also applied many new technologies. Continuously improve and optimize the process to make a series of processes such as software design, testing, and development faster and better.

5.2.3 PERT Chart

And we also use the PERT chart and Gantt Chart for scheduling. This is our PERT chart: Each node in the figure lists the start and end times when we did the task. For clarity, the time required is also listed. The units of time in the figure are weeks, days, and hours.

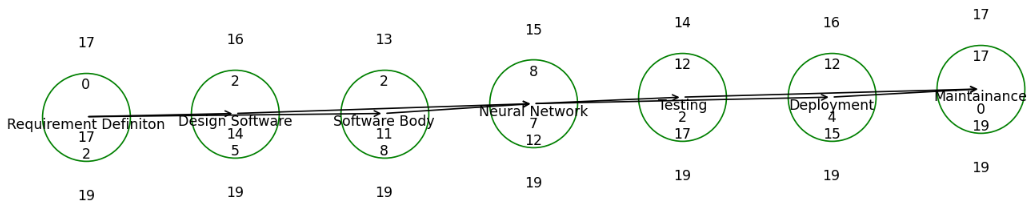


Figure 24: Pert Chart

5.2.4 Gantt Chart

The project lasted 14 weeks in total, divided into 4 sprints. All of these sprints played a very important role in our development process.

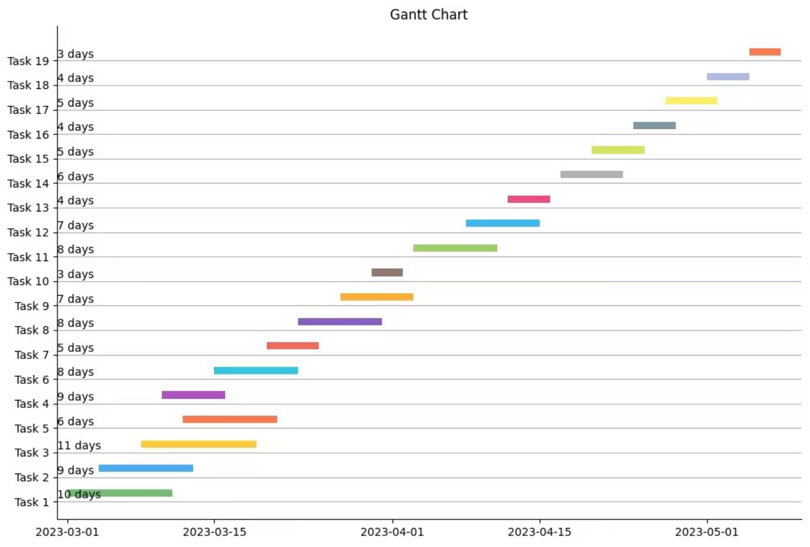


Figure 25: Gantt Chart

5.2.5 Task Network

We have organized a task network based on the software life cycle, including requirements, design, development, testing, and management.

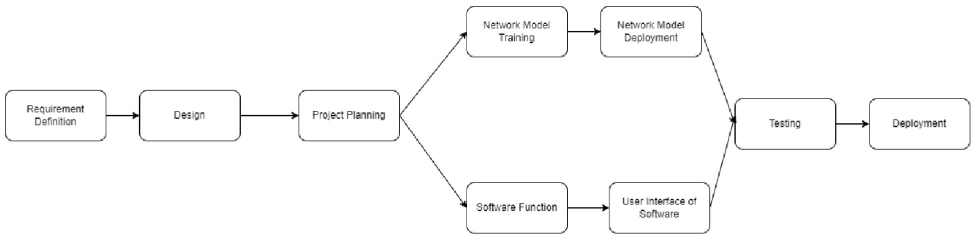
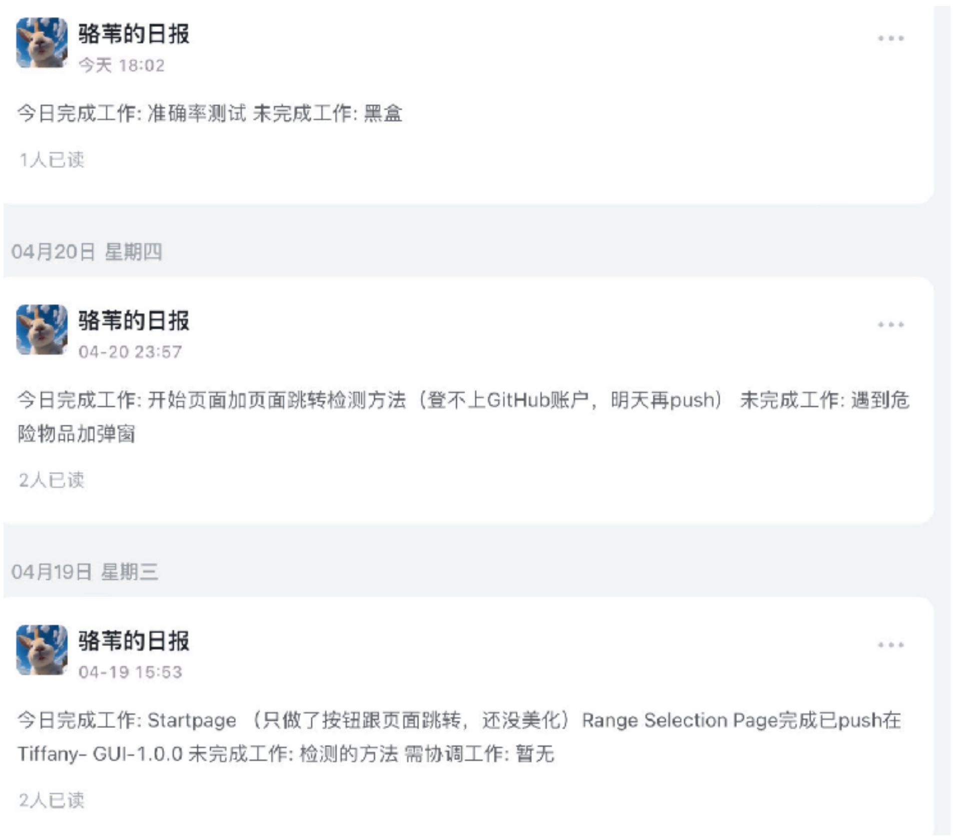


Figure 26: Task Network

5.2.6 Daily Report

We utilize DingTalk as our daily reporting system to assist project managers in overseeing project progress. The following Figure 27 illustrates our daily reporting system.



骆苇的日报
今天 18:02

今日完成工作: 准确率测试 未完成工作: 黑盒

1人已读

04月20日 星期四

骆苇的日报
04-20 23:57

今日完成工作: 开始页面加页面跳转检测方法（登不上GitHub账户，明天再push） 未完成工作: 遇到危险物品加弹窗

2人已读

04月19日 星期三

骆苇的日报
04-19 15:53

今日完成工作: Startpage （只做了按钮跟页面跳转，还没美化）Range Selection Page完成已push在Tiffany- GUI-1.0.0 未完成工作: 检测的方法 需协调工作: 暂无

2人已读

Figure 27: Daily Report

5.2.7 Cost Estimation

Inputs	Tools and Techniques	Outputs
1. Work breakdown structure 2. Activity duration estimates 3. Resource requirements 4. Resource rates 5. Risks	Bottom-up estimating	1. Cost estimates 2. Supporting detail 3. Cost Management Plan

- **Inputs - Work Breakdown Structure&Activity Duration Estimates**

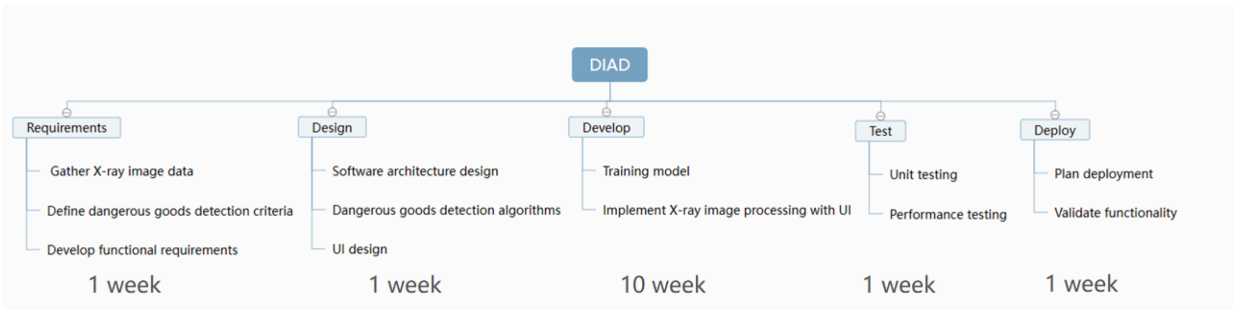


Figure 28: Work Breakdown Structure

Our project revolves around the development of a Dangerous Item Auto Detection(DIAD), designed to identify potential hazards by analyzing X-ray images. The project is structured around a comprehensive Work Breakdown Structure (WBS), featuring five principal tasks: requirements, design, development, testing, and deployment. This ambitious endeavor spans a total duration of 14 weeks.

The initial phase, which involves requirements gathering, is anticipated to be completed within a week. This includes collecting X-ray image datasets of dangerous items, defining detection standards for hazardous goods, and outlining the functional requirements.

The ensuing design stage is also projected to last for one week, during which we will focus on software architecture design, the selection of suitable algorithms, and user interface design.

Subsequently, the development phase is estimated to span ten weeks, encompassing the training of the model and the implementation of the graphical user interface.

In the testing stage, we will be employing unit tests and performance tests to ensure the robustness and efficiency of the developed system.

The final phase, deployment, will consist of planned deployment and functional verification processes, all of which are scheduled to be accomplished within one week.

- **Inputs - Resource Requirements**
  - Human resources: 3 project members

        - including 1 project manager
        - 2 software development engineers.
    - Material resources:
        - software development tools
        - hardware equipment, etc.

- **Inputs - Resource Rates**
    - Human resources rates:
        - ¥1500 per day for project managers
        - ¥1450 per day for software development engineers.
    - Material resource rate:
        - ¥150000 for GPU workstations for training the AI model
        - ¥100000 for X-ray images and data annotation and labeling
        - ¥8000 (¥24000) for each computer
        - ¥100000 for Testing equipment

- **Inputs-Risks**
    - Inadequate data for training and testing the detection algorithms
    - The incompatibility of different operating system environments
    - Longer-than-expected development and testing phases

- **Tools and Techniques - Bottom-up estimating & Outputs**
    - Requirements: (2 developers * ¥1450/d * 5ds) + (1 PM * ¥1500/d * 5ds) = ¥22,000
    - Design: (2 developers * ¥1450/d * 5ds) + (1 PM * ¥1500/d * 5ds) = ¥22,000
    - Development: (2 developers * ¥1450/d * 45ds) + (1 PM * ¥1500/d * 45ds) = ¥42,750
    - Testing: (2 developers * ¥1450/d * 5ds) + (1 PM * ¥1500/d * 5ds) = ¥22,000
    - Deployment: (2 developers * ¥1450/d * 5ds) + (1 PM * ¥1500/d * 5ds) = ¥22,000
    - **Total cost: ¥708,000 + 708,000 * 10%**

- **Outputs - Cost Management Plan**
    - Monitor and control project costs by tracking actual expenses against estimates
    - Establish a change control process for handling scope changes and their impact on costs
    - Conduct periodic risk assessments to identify and mitigate potential cost overruns

5.2.8 PNR Curve

$$
\begin{aligned}
t_0 &= 6 \text{ months} \\
E &= \frac{L^3}{P^3 t^4} \\
E_0 &= \frac{14.1k^3}{2500^3 \times 6^4} \\
&= 0.8 \\
t_a &= 3 \text{ months} \\
E_a &= \frac{14.1k^3}{2500^3 \times 3^4} \\
&= 6.64 \\
t_d &= F \times \text{Effort}^{0.33} \\
&\approx 4.36
\end{aligned}
$$

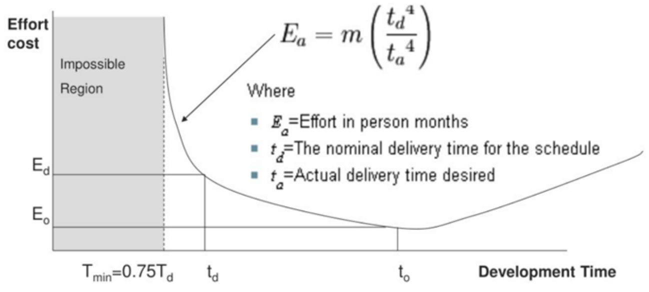


Figure 29: PNR Curve

Based on the analysis of the PNR curve, we have determined that our project can be completed within the anticipated timeframe of 3 months. The PNR curve provides valuable insights into the relationship between effort and delivery time, allowing us to make informed decisions regarding project scheduling. By considering the optimal balance between resources and time, we can allocate the necessary effort to ensure the successful and timely completion of the project. The findings from the PNR curve analysis give us confidence in our ability to meet the project goals within the specified timeframe, providing a solid foundation for effective planning and resource allocation.

5.2.9 Earned Value Management

5.2.9.1 Budget Cost at Completion

The cost of the entire project is categorized into four categories: Software Development, Data Collection, and Prepossessing, Hardware and Cloud Infrastructure , and Testing and Validation. Each category is divided by some items. Table 6 is the budget cost at completion for each item.

Cost Category	Item	Cost
Software Development	Requirements analysis	50,000
	Quality assurance	50,000
	Software development and implementation	600,000
	Documentation and user training	10,000
Data Collection and Prepossessing	X-ray images of dangerous goods	10,000
	X-ray images of non-dangerous goods	10,000
	Data annotation and labeling	20,000
Hardware and Cloud Infrastructure	GPU workstations for training the AI model	15,000
	Storage and backup solutions	5,000
	Cloud computing services	5,000
Testing and Validation	Testing Equipment	10,000
	Testing personnel	15,000
	TOTAL ACTUAL COST=800,000	

Table 6: Budget Cost at Completion

5.2.9.2 Budgeted Cost for Work Scheduled

We have divided our construction period into 14 weeks, and we have allocated a planned budget for each stage to achieve the required workload.It showed in Table 7

State	Period	Planned Workload	BCWS
Requirement	Week 1	(10,000+10,000)+50,000+20,000=90,000	90,000
Design	Week2	50,000	140,000
Development	Week 3-12	600,000+15,000+5,000+5,000=625,000	765,00
Testing	Week 13	10,000+15,000=25,000	790,000
Deployment	Week 14	10,000	800,000

Table 7: Budgeted Cost for Work Scheduled

As of 12:00 AM on Friday of the 14th week of the construction period, we have expended a budget of 795000. The remaining budget of 5000 is allocated for the configuration of cloud services, which is still in progress.

$$\begin{aligned} PercentComplete &= \frac{BCWP}{BAC} \\ &= \frac{795,000}{800,000} \\ &= 0.994 \end{aligned}$$

5.2.9.3 Actual Cost for Work Performed

We record the actual cost of each item from project initiation to project completion in Table 8.

Cost Category	Item	Cost
Software Development	Requirements analysis	70,000
	Quality assurance	50,000
	Software development and implementation	520,000
	Documentation and user training	10,000
Data Collection and Prepossessing	X-ray images of dangerous goods	8,000
	X-ray images of non-dangerous goods	8,000
	Data annotation and labeling	21,000
Hardware and Cloud Infrastructure	GPU workstations for training the AI model	20,000
	Storage and backup solutions	5,000
	Cloud computing services	5,000
Testing and Validation	Testing equipment	10,000
	Testing personnel	5,000
	TOTAL BUDGETED COST=822,000	

Table 8: Actual Cost for Work Performed

We input the data into a formula to calculate cost performance indicators and cost variances, and the results indicated that our project was exceeding its budget.

$$\begin{aligned} Cost\ Performance\ Indicator &= \frac{BCWP}{ACWP} \\ &= \frac{795,000}{822,000} \\ &= 0.97 \end{aligned}$$

$$\begin{aligned} Cost\ Variance &= BCWP - ACWP \\ &= 795,000 - 822,000 \\ &= -27,000 \end{aligned}$$

We also computed the schedule variance, and progress performance indicators, and assessed the progress of the entire project. The findings indicate that the project is currently experiencing a slight delay.

$$\begin{aligned} Schedule\ Variance &= BCWP - BCWS \\ &= 795,000 - 800,000 \\ &= -5000 \end{aligned}$$

$$\begin{aligned} Schedule\ Performance\ Indicator &= \frac{BCWP}{BCWS} \\ &= \frac{795,000}{800,000} \\ &= 0.994 \end{aligned}$$

5.3 Software Maintenance

There are roughly four types of software maintenance activities: corrective maintenance, adaptive maintenance, perfection maintenance, and preventive maintenance.

Corrective maintenance is to correct some potential program errors or design defects exposed under specific use conditions.

Adaptive maintenance is modifying software to adapt to changes in the data environment or processing environment during software use.

Perfection maintenance is to modify the software to incorporate these requirements into the software after users and data processors propose requirements for improving existing functions, adding new functions, and improving the overall performance when using the software.

Preventive maintenance is to improve the maintainability and reliability of software.

Advanced software engineering methods are used in advance to design, prepare and test the software to be maintained or a part (re) of the software, to lay a good foundation for further software improvement.

5.3.1 Corrective Maintenance

We realized that although we have adopted an active testing strategy, it is inevitable that errors may occur in the future. Therefore, when new problems are reported, we will also actively respond, and timely arrange developers to locate and repair problems, to make the system more perfect.

5.3.2 Perfection Maintenance

We considered the problem of code decoupling at the early stage of system design. We split complex logic into small logic. The low coupling of code also means that when we receive new requirements, we can develop new requirements more quickly without affecting the original logic composition, which greatly reduces the cost of our completeness maintenance.

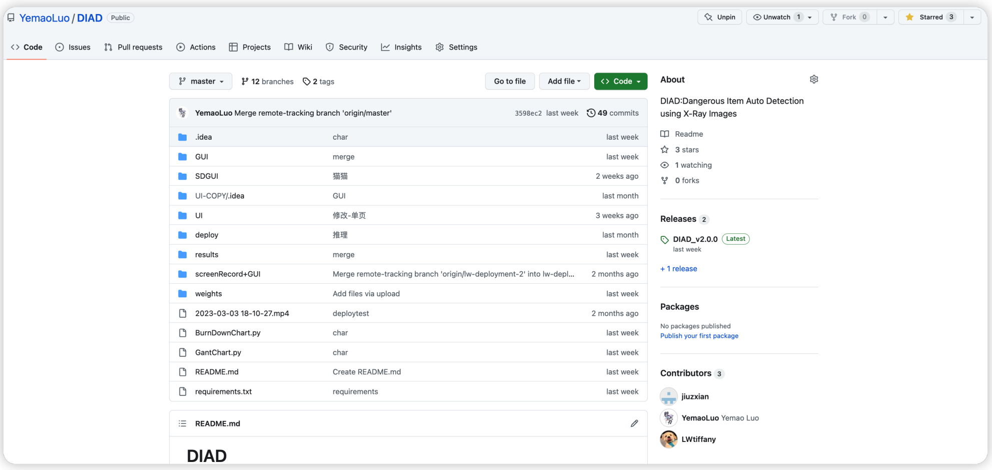
5.3.3 Code Management

GitHub is a popular choice for code management for several reasons. Here are some potential reasons why we choose GitHub as our code management repository:

1) Collaboration: GitHub allows for easy collaboration between developers, as multiple people can work on the same code branch simultaneously. This can be especially useful for larger projects or remote teams.

2) Version control: GitHub provides version control, which allows developers to keep track of changes to the code branch over time.

3) Release: GitHub provides a release function that allows us to release our work version by version that matches our sprints.



YemaoLuo Merge remote-tracking branch 'origin/master' 3598ec2 last week 49 commits

.idea	char	last week
GUI	merge	last week
SDGUI	猫猫	2 weeks ago
UI-COPY/.idea	GUI	last month
UI	修改-单页	3 weeks ago
deploy	推理	last month
results	merge	last week
screenRecord+GUI	Merge remote-tracking branch 'origin/lw-deployment-2' into lw-depl...	2 months ago
weights	Add files via upload	last week
2023-03-03 18-10-27.mp4	deploytest	2 months ago
BurnDownChart.py	char	last week
GantChart.py	char	last week
README.md	Create README.md	last week
requirements.txt	requirements	last week

README.md

DIAD

About

DIAD:Dangerous Item Auto Detection using X-Ray Images

Readme
3 stars
1 watching
0 forks

Releases 2
DIAD_v2.0.0 Latest
last week
+ 1 release

Packages
No packages published
Publish your first package

Contributors 3
jiuzxian
YemaoLuo Yemao Luo
LWtiffany

Figure 30: GitHub Repository for DIAD

5.3.4 Document Management

There are several reasons why LaTeX is a popular choice for collaborative document editing:

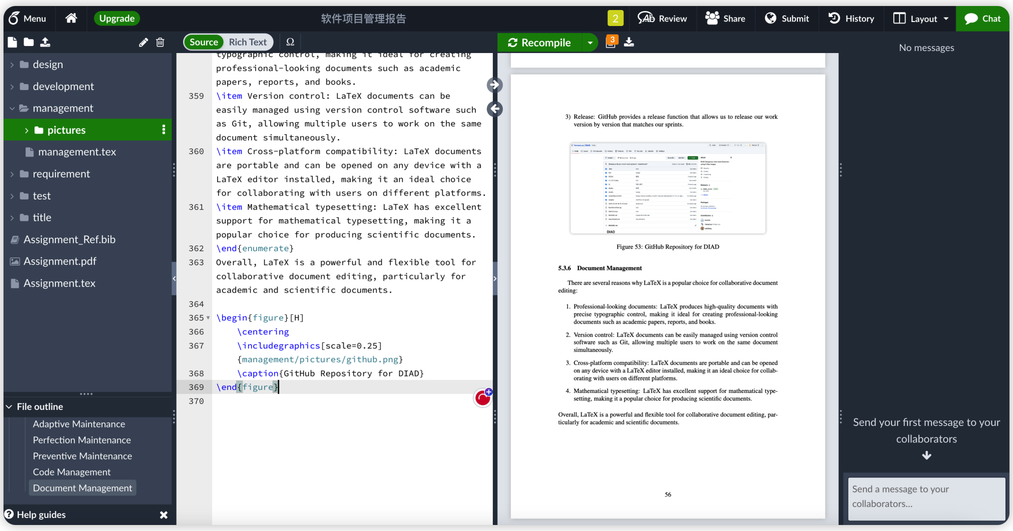
1) Professional-looking documents: LaTeX produces high-quality documents with precise typographic control, making it ideal for creating professional-looking documents such as academic papers, reports, and books.

2) Version control: LaTeX documents can be easily managed using version control software such as Git, allowing multiple users to work on the same document simultaneously.

3) Cross-platform compatibility: LaTeX documents are portable and can be opened on any device with a LaTeX editor installed, making it an ideal choice for collaborating with users on different platforms.

4) Rich typesetting: LaTeX has excellent support for mathematical typesetting and display charts and figures, making it a popular choice for producing scientific documents.

Overall, LaTeX is a powerful and flexible tool for collaborative document editing, particularly for academic and scientific documents.



```
typographic control, making it ideal for creating
professional-looking documents such as academic
papers, reports, and books.
\item Version control: LaTeX documents can be
easily managed using version control software such
as Git, allowing multiple users to work on the same
document simultaneously.
\item Cross-platform compatibility: LaTeX documents
are portable and can be opened on any device with a
LaTeX editor installed, making it an ideal choice
for collaborating with users on different platforms.
\item Mathematical typesetting: LaTeX has excellent
support for mathematical typesetting, making it a
popular choice for producing scientific documents.
\end{enumerate}
Overall, LaTeX is a powerful and flexible tool for
collaborative document editing, particularly for
academic and scientific documents.

\begin{figure}[H]
    \centering
    \includegraphics[scale=0.25]
    {management/pictures/github.png}
    \caption{GitHub Repository for DIAD}
\end{figure}
```

3) Release: GitHub provides a release function that allows us to release our work version by version that matches our sprints.

Figure 53: GitHub Repository for DIAD

5.3.6 Document Management

There are several reasons why LaTeX is a popular choice for collaborative document editing:

1. Professional-looking documents: LaTeX produces high-quality documents with precise typographic control, making it ideal for creating professional-looking documents such as academic papers, reports, and books.
2. Version control: LaTeX documents can be easily managed using version control software such as Git, allowing multiple users to work on the same document simultaneously.
3. Cross-platform compatibility: LaTeX documents are portable and can be opened on any device with a LaTeX editor installed, making it an ideal choice for collaborating with users on different platforms.
4. Mathematical typesetting: LaTeX has excellent support for mathematical typesetting, making it a popular choice for producing scientific documents.

Overall, LaTeX is a powerful and flexible tool for collaborative document editing, particularly for academic and scientific documents.

56

Figure 31: Overleaf Project for DIAD

5.4 RMMM Plan Using RIS

RMMM Plan Using RIS deployed as web page: https://yemaoluo.github.io/DIAD/RMMMPlan.html

Risk ID: R001

Date: March 1, 2023

Probability & Impact: High Probability, High Impact

Description: There is a possibility that our x-ray detection system may not detect certain dangerous items, which could pose a risk to the safety of people and property.

Refinement/context: We will need to conduct extensive testing and analysis to identify potential blind spots in the system and ensure that all possible dangerous items are properly detected.

Mitigation/monitoring: To mitigate this risk, we will use a combination of human supervision and advanced algorithms that can identify potentially dangerous items. We will also set up a monitoring system that will alert us to any potential issues or abnormalities in the detection process.

Risk ID: R002

Date: August 1, 2023

Probability & Impact: Medium Probability, Medium Impact

Description: There is a risk that the software used in the x-ray detection system may affect the performance of the machine, leading to a delay or even a complete shutdown of the system.

Refinement/context: To mitigate this risk, we will conduct extensive testing and analysis of the software to ensure it is compatible with the machine and does not adversely affect its performance.

Mitigation/monitoring: We will monitor the performance of the machine and conduct routine checks to ensure the software is functioning properly. Additionally, we will have a backup system in place in case of any unexpected issues or shutdowns.

Risk ID: R003

Date: April 6, 2023

Probability & Impact: Medium Probability, Medium Impact

Description: There is a risk that team members responsible for the x-ray detection system may experience illness or inadequate technical ability, leading to delays and potential errors in the detection process.

Refinement/context: To mitigate this risk, we will ensure that all team members receive proper training and have clear guidelines for their responsibilities. We will also establish a backup plan in case team members are unable to perform their duties.

Mitigation/monitoring: We will provide ongoing training and support for team members and regularly monitor their performance. We will also establish a system for team members to communicate any potential issues or concerns.

Risk ID: R004

Date: April 9, 2023

Probability & Impact: Moderate Probability, High Impact

Description: There is a risk that inadequate performance may prevent real-time monitoring, causing delays and inconvenience to personnel accessing the security mechanism.

Refinement/context: To address this risk, we will use a lightweight monitoring model that balances performance and accuracy. We will conduct benchmark testing on various monitoring models and select the one that best meets our requirements.

Mitigation/monitoring: We will regularly monitor the performance of the monitoring system and conduct testing to identify and address any potential issues before they can impact operations. In case of any issues, we will immediately notify our technical team and implement our contingency plan.

Risk ID: R005

Date: April 20, 2023

Probability & Impact: Low Probability, Low Impact

Description: There is a risk that the withdrawal of capital or resources by the contracting party may result in insufficient funds and resources to complete the development, leading to delays or even project failure.

Refinement/context: To address this risk, we will include clauses in the contract stipulating the contracting party's investment obligations and exit mechanisms. We will also conduct regular financial assessments to determine if sufficient funds are available to meet project requirements.

Mitigation/monitoring: We will closely monitor the financial status of the contracting party and conduct regular assessments to ensure that adequate funds and resources are available to meet project needs. If necessary, we will take necessary action to prevent project delays or failures.

6 Acknowledgments

We would like to express our sincere gratitude to Professor Zhang Tao for his invaluable guidance and support throughout this semester. His expertise and insights in software project management have been instrumental in shaping our understanding of this subject. His lectures, feedback, and suggestions have inspired us to think critically and have helped us develop a strong foundation in this field. We are grateful for the effort and time he has invested in teaching and guiding us. We are fortunate to have had the opportunity to learn from him and will carry his teachings with us throughout our academic and professional journey. Thank you, Professor Zhang Tao, for your exceptional guidance, encouragement, and mentorship.

We would also like to express our heartfelt appreciation to our group members for their tireless efforts and unwavering support throughout this project. Our collaboration has been an incredibly pleasant experience, and we could not have asked for a better team. Each member's unique perspectives and contributions were critical in shaping our project's success. We are grateful for the dedication, hard work, and commitment of each group member, which made this project possible. Working with such an outstanding team has been a privilege, and we are proud of what we have accomplished together. Thank you all for your support and for making this a memorable and enjoyable experience.