TED UNIVERSITY

CMPE492

Computer Engineering

Library Occupancy Detector

Test Plan Report

30.11.2024

Team Members

Burak Koç

Buse Öner

Furkan Safa Altunyuva

Table of Contents

1. Introduction

The tests of our Library Occupancy Detection project should consist of many different tests and test environments, as they cover both the user-side library management and students who want to learn the occupancy rate. In addition to users, the model's overall accuracy rate and the test of its ability to detect each object that can be placed on the table with appropriate accuracy rates separately are also very important as they affect the overall project. Since object detection is at the core of our project, the model and the project should be tested frequently to ensure that it evaluates various lighting, camera focuses, and clothes worn due to different weather conditions in the most accurate way, and the increase in the model and project's accuracy rate should be examined.

Since the system we plan to build is aimed at long-term use, we need to plan how these tests will yield results in possible future scenarios while creating the tests and developing the tests by taking these situations into account. For example, if students understand that a type of object cannot be detected by our system, the system will not detect that table as a “hold” when that object is used to hold space and will incorrectly update the database. In order to prevent these situations, we plan to conduct regular tests even after the project is developed and put into use. We plan to quickly detect and fix some problems that users may abuse with the tests we will conduct at certain intervals.

Since we plan to conduct tests on a regular basis, we try to keep the tests we will conduct during the development phase as generic and dynamic as possible so that these tests can be used after the product is developed with minimal changes and can even be automated if possible.

1.1 Definitions, acronyms, and abbreviations

- KVKK: Personal Data Protection Law, the Turkish legislation governing the protection of personal data, in accordance with Kişisel Verilerin Korunması Kanunu.
- myTEDUPortal: The website where students, faculty, and staff can access academic and administrative services, such as course registration, grades, and schedules.
- TeduAPP: The "Tedu App" is an application designed and developed by Tedu students, aiming to guide students about school activities, class schedules, and communities within the school under the slogan of "by students, for students.
- Occupancy Rate: The ratio of occupied seats to the total seating capacity in the library, basically representing the percentage of students currently utilizing seats.
- Hold Status: A situation where a seat or table is considered full due to the presence of personal items left by a student.
- YOLOv10: You Only Look Once, is an object identification method known for its high accuracy and real-time detection capabilities.
- Real-Time Detection: The ability of the system to process/analyze and show data almost instantly.
- QA: Quality assurance is defined as a component of quality management that aims to provide assurance that quality requirements will be fulfilled.

2. Scope

Although we have done many tests so far, we plan to do 3 types of tests in the current phase of our project.

2.1 Tests on Applications Used by Library Staff

The purpose of this test is to identify problems and errors that library staff may encounter in the interface we developed. In the meantime, we plan to get feedback from testers about user experience and ease of use.

2.2 Data Communication Tests with Third Party Applications

The purpose of this test is to test the ability to connect seamlessly with third-party applications such as TEDU App and myTEDUPortal. By doing these tests, we hope to see possible errors and areas that can be improved while sharing instant library occupancy rates with third-party applications in a reliable and fast way.

2.3 Model Accuracy Rate Tests

The purpose of this test is to test whether the YOLO10n and YOLO10x models we currently use can achieve the desired level of accuracy with the current camera angles and images. If this test is insufficient, we plan to use other models or train them with new data, and if it is sufficient, we plan to use these models or try updates with YOLO11.

3. Quality Objectives

Although we determined the primary and secondary objectives of the tests, the importance and the order of the planned tests may change due to the time limitations and the requests of the library staff. Any major changes that may affect the structure of the project will be changed in the other documents as well. Possible changes will be notified to the supervisor, juries, and the library staff.

3.1 Primary Objectives

The primary priority of our tests is to ensure that our system (except for very specific scenarios) can correctly detect users and the objects they place on their desks or chairs to take up space.
In addition to the model’s level of accuracy, our project’s communication with third-party applications and APIs is very important for the purpose of the project.

3.2 Secondary Objectives

Error tests and user experience tests performed on the application to be developed for library staff can be classified as secondary objectives of our overall tests. Although we want the operating systems to use a problem-free and easy-to-use application, we do not plan to take on these features before the core features of our project are tested in the most accurate way. This situation is open to change as a result of the directives of the management, which is the client-side policy of our project.

4. Test Approach

Within the scope of our Library Occupancy Detector project, a comprehensive testing process has been planned to verify that each component of the system works reliably and consistently. This process is implemented in the library environment with the help of developers and covers the areas mentioned above. To further detail these areas, the application tests developed for library staff aim to test the functionality and user experience of the application that library staff will use. In this context, user login/logout transactions, real-time data display, notification and reporting features are verified to work correctly.

With Object Detection Model Tests, the object detection model of our system is tested under various scenarios within the library (different lighting conditions, camera angles and human density). The accuracy of the model is optimized by analyzing the result rates of the machine learning metrics (such as true negative, true positive) required for testing accuracy.

As we mentioned in the document, our system is dependent on third party APIs for data exchange. To ensure that API integrations work correctly, tests are performed on timeouts, data mismatches, and update statuses.

All these testing processes are carried out in the library environment as close as possible to real usage conditions and are supported by the observations of the developers. Additionally, necessary precautions are taken against possible problems within the scope of KVKK (Personal Data Protection Law) during the test processes. User privacy and consent are protected by ensuring that cameras shoot from above and do not show human faces, obtaining permission from users during the test and only processing necessary information. Security tests are carried out at different layers of the system for data security and the results are reviewed.

4.1 Test Automation

Although no test automation has been developed at the current stage of our project, tests will be planned as regularly and in layers as possible so that tests can be easily automated in the future.

5. Entry and Exit Criteria

5.1 Entry Criteria

To start the test phase, the following conditions must be met

- The test environment has been set up, including the necessary hardware, lighting variations, and object configurations
- Test data has been prepared and verified, covering diverse scenarios such as object types, lighting conditions, and user interactions.
- A comprehensive set of documentation, including functional specifications and API documentation, is provided to the development team.
- QA testers conduct research on the functionality and testing procedures of the system and develop a comprehensive understanding of the quality assurance system and plan the testing approach accordingly.
- A trial has demonstrated the capability to connect seamlessly with third-party applications, including the TEDUApp and the myTEDUPortal, with successful data transfer to Firebase DB for accessibility via the specified third-party applications.

5.2 Exit Criteria

The testing phase will be deemed to have reached a satisfactory conclusion and be ready for production deployment when:

- All critical functionalities meet the performance benchmarks, including object detection accuracy and response time.
- No unresolved critical or high-severity bugs remain.
- Regression tests confirm that the implemented fixes have not introduced new issues.
- User Acceptance Testing (UAT) has been successfully completed with approval from library staff.
- The integration with third-party applications has been verified, ensuring seamless data exchange, with data successfully accessible via TEDUApp and myTEDUPortal.

6. Suspension Criteria and Resumption Requirements

6.1 Suspension Criteria

The following circumstances will result in the suspension of testing:

- **In the event of severe defects in the system:** The build contains critical issues that significantly impede or restrict the progress of testing. These include major flaws in the object detection functionality and system integration failures.
- **Modifications to the specifications:** Significant alterations to the project requirements, as proposed by stakeholders (e.g., library staff, or university administration), which impact the scope or objectives of the tests.
- **Issues pertaining to the software or hardware**: In addition, the test environment may present challenges, such as hardware failures (e.g., cameras or servers) or software issues (e.g., malfunctioning detection algorithms, issues with third-party application integration).
- **In the event of unavailability of the requisite resources:** If the designated quality assurance personnel are unable to continue with the testing process due to unexpected circumstances, such as illness or delays in resource allocation (e.g., access to test data or systems), the project may encounter delays.

6.2 Resumption Criteria

The resumption of testing will only be permitted once the issues that led to the suspension have been satisfactorily addressed and resolved. The identified issues can be as listed as following:

- It is not possible to commence the testing phase until the critical flaws have been rectified. It is imperative that any significant deficiencies that impede the advancement of testing are promptly rectified. This encompasses all issues pertaining to object identification precision, response times, and integration.
- It is of the upmost importance to ensure that the requirements are precise and unambiguous. Furthermore, it is essential to guarantee that the testing procedure aligns with the revised objectives, following the necessary modifications to the requirements and the appropriate recording of these changes.
- The objective is to stabilise the testing environment. To ensure that the test environment is fully functional, it is necessary to address any hardware or software issues that have caused delays.
- Once the requisite resources, such as personnel and system access, have been reinstated and made available, testing will resume.

7. Test Strategy Plan

7.1 QA Role in Test Process

The quality assurance (QA) process is a crucial element in the development of any product or service, as it enables the early detection of potential issues and the prevention of flaws. This not only ensures the delivery of a high-quality product that meets the expectations and requirements of the end user but also reduces costs by minimizing the need for rework. Furthermore, a robust QA system provides consumers with confidence in the dependability of the product or service and ensures compliance with industry standards, thereby mitigating legal risks. In summary, a comprehensive QA process is essential for achieving both functional and strategic project goals.

The quality assurance process entails the establishment of standards, the implementation of checks, and the validation of outcomes at each stage of the development process. It encompasses a range of activities, including test planning, test case creation, execution, defect tracking, and reporting.

7.1.1 Defining Standards

It costs more to remedy a bug detected during testing than to prevent it during the requirements design phase. As a result, QA specialists must be involved in the study and definition of both functional and non-functional software requirements. Basically, the requirement specifications are provided by the customer and the QA team is responsible for understanding and analyzing these requirements.

The library occupancy detection system requires precise recognition of people and items inside predefined Regions of Interest (ROIs). It must handle video streams in real time and work flawlessly with YOLOv10, OpenCV, and Firebase for data storage and visualization. The system needs to be simple to use, adaptable, and capable of handling many streams simultaneously. It also needs to exceed stakeholder expectations by providing accurate and simply available occupancy data, ensuring the system's efficiency and effectiveness in real-world applications

7.1.2 Test Planning

In test planning, insights from the requirements analysis phase are used to determine the testing approach, scope, budget, and deadlines. The plan covers the various types and degrees of testing, bug tracking methods and tools, and resource allocation.

The test cases will comprehensively cover all scenarios required for detecting library occupancy. This includes:

Interface Usability for Library Staff: The interface is intuitive, user-friendly, and error-free, allowing library staff to quickly execute their duties and track occupancy rates.

Third-Party Application Integration: Ensure uninterrupted data communication and share instant library occupancy rates reliably and quickly with applications such as TEDU App and myTEDUPortal.

Object Detection Accuracy: Validating the system's ability to detect people and objects (e.g., chairs, phones) with high precision.

These detailed plans will guide the testing process to ensure robust system functionality and reliability.

7.1.3 Test Matrix Creation

The QA team will construct a Test Matrix that associates each test case with the associated requirement. This matrix is used as a tool to guarantee that all criteria are addressed by specific test cases, allowing for explicit traceability between requirements and tests. Using the Test Matrix, the QA team can ensure that no requirements are overlooked and that all have been thoroughly tested. This systematic approach provides comprehensive testing coverage and makes it easier to identify deficiencies in the testing process. The following test matrix illustrates the correlation between test cases and their corresponding requirements.

Test Case ID	**Requirements**	**Test Description**	**Expected Outcome**
TC-01	Object Detection Accuracy	Test the YOLO10n and YOLO10x models for occupancy detection in current camera angles, lighting conditions and clarity levels. This includes human detection and the detection of objects such as coats and bags used to hold an area.	Models achieve a minimum threshold of accuracy (e.g. $\geq$90%) for detecting people and objects in different scenarios.
TC-02	Object Detection Accuracy	In the event of YOLO10n/YOLO10x failure, test alternative models or retrain models with updated data.	New or retrained models meet the required accuracy threshold.
TC-03	System Extensibility (Handling Multiple Feeds)	Test the system's behavior when processing multiple camera feeds.	The system is capable of processing multiple feeds in a simultaneous and uninterrupted manner.
TC-04	Database Validation	By verifying the Firebase integration, it is checked that the occupancy data is saved correctly and accessible.	The system accurately saves occupancy data to the Firebase database, ensuring the absence of errors or inconsistencies in the retrieval process and shares the data with third-party applications.

TC-05	Library Staff Interface Usability	Verify that the interface is intuitive to navigate, error-prone, and easy to use. In addition, data retrieval from the database must be reliable and fast.	Library staff will not be confused by the interface and all features will work the way as expect. Also, data is displayed correctly with no latency.
TC-06	Third-Party Data Communication (TEDU App)	For instant occupancy updates, verify data sharing with the TEDU App.	Data is shared accurately and timely, without any delays or inconsistencies.
TC-07	Third-Party Data Communication (myTEDUPortal)	The reliability of communication can be tested with myTEDUPortal under a variety of load conditions.	The system is capable of processing data requests continuously and efficiently without any noticeable delays or failures.

7.1.4 Reviewing Matrix and Test Cases

During this process, the test cases and test matrix are peer reviewed by a designated QA manager. Depending on the feedback received as a result of the review, the QA person who created the relevant test case should make the necessary updates. After resubmitting the updated versions for approval, the reviewer performs a final check to make sure all recommendations have been taken into account and the test material is prepared for use.

7.1.5 Test Creation

At this stage, it is necessary for the quality assurance (QA) teams to develop comprehensive test cases and checklists that encompass the full range of requirements for the software. To achieve this, the test matrix will be used as a reference tool. The obtained outcomes should be compared to the desired results, and the system should be improved accordingly.

7.1.6 Test Execution

During the test execution phase, all critical functions of the system are tested in accordance with the specified test scenarios and test matrix. Initially, unit tests are conducted by developers to ascertain the functionality of the system's fundamental components. Subsequently, manual tests are performed on comprehensive test scenarios.

The testing phases encompass the following elements:

- Verification of the accuracy of the detection of objects
- Assessment of the processing capabilities of the camera feed
- Validation of database integration for accurate data storage and retrieval
- Evaluation of the usability of the library staff interface
- Ensure data communication with third-party apps is reliable

7.1.7 Retesting and Regression Testing:

The errors identified during the test are subjected to analysis and correction in accordance with a pre-established priority order. Once the underlying causes of the errors have been identified and resolved, the relevant tests are repeated to ensure that the system produces the anticipated results.

7.1.8 Completion of the Testing Phase

Once all identified errors have been resolved and the required tests have been successfully re-run, the test phase can be considered complete. If the system is found to meet all specified requirements and perform in accordance with expectations, the testing process is terminated, and the system is confirmed to be ready for use.

7.2 Bug Life Cycle

The bug life cycle covers all the stages that an error goes through until it is detected and resolved. The process commences with the reporting of the error and culminates in its resolution by assigning it to the pertinent team. Once a solution has been implemented, the error is tested and verified. If the error has been successfully rectified, the process is concluded. However, if the error reoccurs, the process is restarted. This cycle is essential for enhancing the quality of the software, resolving errors in a timely manner and guaranteeing the reliability of the product. In order to optimize the use of limited time, it is essential to resolve errors in accordance with a defined priority order and to avoid unnecessary stages.

7.3 Testing Types

Black Box Testing

This type of test focuses on the functional requirements of the software. With various data sets from the user, it is tested how the system performs certain functions and requirements. For example, functions such as whether the library occupancy system detects people and objects correctly, whether data is stored and transmitted correctly, are tested.

Unit Testing

Unit testing is a method of testing individual components of a system in isolation to ensure that they function as expected. Unit testing represents the initial stage of software testing, preceding integration testing. Unit tests are developed and executed by the developers themselves. In the case of the library occupancy detection system, unit testing serves to verify the accuracy of detection algorithms, ensure the smooth operation of the database for storing and retrieving data, validate the functionality of user interface (UI) elements, and confirm the proper integration of camera feeds. By identifying and resolving issues at an early stage, unit testing enhances system quality and establishes a foundation for successful integration and system testing.

Integration Testing

Integration testing is a type of software testing that involves evaluating the functionality and integration of multiple modules or components within a software application. It assesses whether the individual components operate as intended and are integrated correctly. For instance, integration testing for the YOLOv10 model entails verifying that the model processes camera images correctly and facilitates data transfer to the Firebase database. Additionally, it ensures that the application shares data with third-party applications, such as the TEDU App and myTEDUPortal, in an appropriate manner.

System Testing

System testing is a comprehensive type of test that verifies whether both functional and non-functional requirements are met by evaluating the software as a whole. This process entails a comprehensive examination of the software's behavior in a multitude of usage scenarios and conditions, encompassing both normal and unexpected situations. For instance, it is confirmed that students are able to accurately display library occupancy rates through applications such as TEDUApp and myTEDUPortal, and that data can be transferred in its entirety. Furthermore, the ability of the library staff to monitor current and past occupancy rates through the developed interface and to retrieve these data from the database without any interruption or delay is being evaluated. Additionally, the system's ability to perform stably over an extended period or in scenarios involving multiple camera connections is assessed. These and many other tests like them ensure that the software meets user expectations and delivers reliable performance.

Performance Testing

The performance test is designed to assess the system's behavior when faced with a significant workload. In the context of the library occupancy system, this implies an examination of the ability to process multiple camera feeds in a simultaneous manner, as well as an investigation into the system's capacity to manage high user traffic. Furthermore, the assessment encompasses the evaluation of optimal performance criteria, such as page load time.

User Acceptance Testing

The User Acceptance Test (UAT) is conducted to ascertain whether the system meets the designated user requirements and is fit for operational use. This test is carried out to enable library personnel and students to evaluate the functionality of the system and identify potential usage issues. The UAT process typically encompasses two stages: Alpha Test and Beta Test.

Alpha testing

Alpha testing represents an internal phase during which developers evaluate the program to identify and address critical issues before external testing begins. In the context of the library occupancy detection system, this entails confirming the precision of the YOLOv10 model, validating Firebase data integration, and ensuring seamless integration with third-party applications such as TEDUApp. Any deficiencies identified during alpha testing are resolved before proceeding to beta testing or release.

Beta testing

Beta testing can be defined as a specific type of evaluation performed by a group of external users who do not belong to the development team. The primary objective of such a test is to gather user feedback and identify any issues that may have been overlooked during alpha-testing. Additionally, beta-testing assesses the system's stability under high-volume use and in diverse settings, such as when connecting numerous cameras simultaneously. The feedback obtained during beta-testing is then employed to implement final modifications and prepare the product for its final release.

7.4 Bug Severity and Priority Definition

The identification of bug severity and priority facilitates the optimization of the problem-solving process, enabling the determination of the impact of errors in the software development process and the establishment of an effective resolution sequence. The classification of errors by severity and urgency allows for a more effective use of time. Correct categorization and prioritization of errors improves time management and reduces the impact of errors on project schedules. A table can be used to provide a clear and comprehensive guide to the types of errors and their respective resolution priorities. This approach facilitates a deeper understanding of the sequencing logic of errors and enhances awareness among team members.

Severity Table

Severity ID	Severity Level	Severity Description	Sample Scenario
1	Critical	It is imperative that such errors are rectified without delay, as they impede users from accessing the software's primary functions and result in the software becoming inaccessible.	The system crashes when trying to retrieve occupancy data from the database, or the library's occupancy status cannot be displayed at all due to data corruption.
2	High	These errors have the potential to disrupt critical functions, yet the system remains operational. While a solution is imperative, it is not an immediate necessity.	In the event of a delay in data transfer, myTEDUPortal and TEDU prevent users from accessing instant occupancy data on the platforms where students access such data.
3	Medium	These errors can often negatively affect the user experience, but they do not have a major impact on the overall functioning of the system.	There is a discrepancy between the real-time camera feed and the occupancy data display, which may temporarily resolve when the camera is restarted.
4	Low	Such discrepancies are typically minor in nature, either visual or textual in character, and do not impact the system's operational availability.	A minor error in the presentation of historical occupancy statistics in the library staff interface, or a minor inaccuracy in the explanation of the system's database structure.

Priority Table

Priority ID	**Priority Level**	**Priority Description**
1	Must Fix	These bugs are urgent and demand quick action because they inhibit the system's optimal operation or cause serious disruptions. It is critical that they be addressed as quickly as possible.
2	Should Fix	These bugs must be addressed with minimal delay since they have the potential to affect significant system components and disrupt user experience or operational flow. While there may be a temporary fix, these concerns must be addressed as soon as feasible.
3	Fix When Have Time	These are essential issues, yet they do not have a significant impact on the system. They can be postponed to prevent delaying the deployment date and are often corrected after high-priority errors have been resolved.
4	Low Priority	These issues have a negligible impact on user and system performance and can be addressed subsequently. It is recommended that these bugs be resolved once all other bugs have been fixed.

8. Resource and Environment Needs

The setting where the software will be tested is referred to as the test environment. For the software to properly meet its requirements and test for any type of error or glitch situation, this needs to have the right hardware, software, and configurations. To ensure the reliability of the test results, it is crucial that the test environment is properly configured.

Test Component	**Requirement**	**Description**
Hardware	Camera, Computer, USB cable	Necessary devices for monitoring library occupancy.
Software	OpenCV, YOLOv10, YOLOv11, Firebase, PyCharm	The software and libraries utilized for the purposes of development and testing.
Network Configuration	Local Network Connections, Internet	It is imperative that the network connections are fully operational and functional throughout the duration of the test.

9. Test Schedule

A document known as the test schedule outlines which tests should be run at each stage of the software development process as well as when they should be run. To ensure that the testing procedures are finished on time, it is imperative that this schedule be followed.

Task Name	**Start Date**	**End Date**	**Status**
Review Requirements	02.11.2024	19.11.2024	Completed
Test Planning	20.11.2024	26.11.2024	Completed
Test Case Design	26.11.2024		In Progress
Preparation and Installation of the Environment	21.10.2024		In Progress
Unit Testing			Not Started
Integration Testing			Not Started
System Testing			Not Started
User Acceptance Testing			Not Started
Final build testing and defect resolution			Not Started
Performance testing			Not Started
Release to Production			Not Started

10.Control Procedure

Control procedures are defined as the methods that are determined in order to ensure that all steps in the testing process are carried out in a consistent and efficient manner. These procedures serve to guarantee that any errors that may occur are duly tracked, that tests are performed correctly, and that the results are duly documented.

Procedure	**Description**	**Application Method**
Error Tracking and Reporting	Recording of errors detected during the test	GitHub
Test Environment Management	Regularly checking of the test environment	The test environment is checked daily, and updates are made.
Version Control	Management of software versions	The new versions offered by Ultralytics are reviewed and the necessary updates are made. The software is subjected to compatibility tests and updates are reflected on GitHub

11.Roles and Responsibilities

In the context of software development and testing, each team member is assigned a specific role and responsibility. These roles serve to ensure the efficient progression of the process and guarantee the accuracy of each step.

Role	**Responsibility**
Project Manager	Creating test plans and managing test processes
UI Developer	To perform user interface tests, correct errors and verify that the interface is working correctly.
Programmer	Creation of unit tests, the provision of support for integration tests, and the resolution of errors within the code base.
QA Staff	To design, implement test scenarios, create reports and verify the performance and reliability of the system.

12.Risks

It is crucial to identify potential risks that may be encountered during the software development process in advance in order to minimize the effects of these risks. Effective risk management contributes to the smooth progress of the project process.

Risk	**Possible Impact**	**Suggested Solution**
Hardware Failures	Failures in the testing process	It is recommended that additional hardware be made available for replacement.
Software Failures	Failure of tests	The utilization of preliminary unit testing and error tracking tools.
Time Constraints	Failure to complete the tests on time	Reorganize the test plan in line with the determined priorities and to use the available resources efficiently by adhering to the plan.
Environmental Factors	Problems in the physical test environment (network, server, etc.)	The objective is to detect environmental problems in advance, to develop solutions to these problems and to create effective intervention mechanisms by providing temporary solutions.

13.References

1. YOLOv10 Algorithm: YOLOv10
2. Software Architecture for Computer Vision: ResearchGate - Software architecture for computer vision: Beyond pipes and filters
3. Proposed ICPS Real-Time Monitoring System: ResearchGate - Architecture of the proposed ICPS real-time monitoring system
4. diagrams.net [For other consumer use]. diagrams.net
5. Real-Time Architecture Considerations: Upsolver Blog - Building a Real-Time Architecture: 8 Key Considerations
6. Types of Testing Guide: How to set up QA processes from scratch | BrowserStack
7. QA Setup Process Guide: Different Types of Testing in Software | BrowserStack