CMPT 276: Group 23

One Direction Ltd.'s

Issue Tracking System

Release 1

Release Date: 2024-06-07

Approved by:

Nooran Chellabi - Project Manager
Vivian Lau - Marketing Dept.
Ashish Kalam - Software Engineering Dept.
Julianne Soriano Diaz - Quality Assurance Dept.

Release History:

Release 1:	2024-06-07 -Software requirements specifications released

Release Table

RELEASE TABLE PAGE						
Configuration Item:	Version	Number of Each Configuration Item in:				
	Rel. 1	Rel. 2	Rel. 3	Rel. 4	Rel. 5	Rel. 5A
Documents:						
Requirements Spec	1					
User Manual						
Architectural Design						
Modules (Main first, then alphabetical):						
GeeWiz.mod						
libA.def						
libA.mod						
libB.def						
libB.mod						
libC.def						
libC.mod						
Other Files						
makefile						
DemoData						

Version History

Version 0 - Original by One Direction Ltd. 2024-06-07

Conventions Key

This page is for any naming/formatting conventions used throughout the document.

Boldface - Used when placing emphasis on important terms, also used for distinguishing between headers of subsections and paragraphs belonging to them.

Underline - Also used for placing emphasis on terms.

Table of Contents

1.0: Introduction

1.1: System Overview

The application area for this software is the tech/software industry, specifically for the software company 5 Software On Systems, or 5SOS for short. 5SOS will use our software to keep track of bugs and features to be added to their own products. There are several types of jobs within this application area, ranging from business related jobs such as marketing to people working tech jobs such as software engineers. This type of software is useful for all types of people working within the software industry. One of the more common issues being run into without this software is the issue of workflow and how much it could be improved by using this software. Without this software, companies commonly have actual sheets of paper that log things such as bugs and new features. Using paper can make it harder to stay organized, thus resulting in a slower workflow.

We are introducing a system that keeps track of bugs, new features, and status updates all in one place. This system can be used by both customers via employees over the phone and internal employees to keep track of changes made (or needed to be made) to the company's products. On the customer side, customers can call in to report a bug they ran into, or to give suggestions for a feature they may find to be helpful in the future. An employee logs the information given to them by the customer, as well as the customer's contact information if it is their first time calling. Customer information is stored in a database for ease of access. Similarly, employees from any department can do the same thing. They are also able to keep track of what needs to be done and can provide status updates on certain features, for example, whether or not a bug has been fixed.

The major functions of the system include:

- The ability to log issues into the issue tracking software.
- The ability to view all issues that have been reported.
- The ability to edit an issue (provide a status update?)
- The ability to store customer and employee information in a database for ease of access in the future.

As mentioned previously, we developed this to improve company productivity. The major functions available in this software allow users to quickly log whatever they find necessary to be addressed in a much quicker fashion than if they had to do it the old-fashioned way i.e. using pen and paper, which would eventually need to be organized. Because of the vast difference between the old and the new system, this software will greatly improve workflow within the application area by getting rid of some of the steps that occur between a complaint being issued and the employee responsible for fixing it.

1.2: Business Objectives

The current market for this software is the software industry itself because it allows engineers to easily find and fix issues within their own software. It also allows customers to easily report issues or new features that the engineers need to deal with. There are already several issue tracking software available in the market, so there is ultimately going to be competition, however, what sets us apart is that this software is tailor-made for our client, meaning there would only be competition if this product was available to the general public. There are plans for this software to be sold to multiple clients in the future, in the event of that going through, some of the features that are anticipated in future releases may be what sets us apart from the competition.

The software we are developing will be sold to the client for $300,000 according to the business contracts that were made. The contracts also go into more detail about ownership and licensing rights in the case of us potentially wanting to sell our software to other clients.

1.3: Software Project Outline

There are many steps that need to be taken before building the actual software so we should start as soon as possible. Since we are developing software for a client, there are necessary deadlines in order for things to go smoothly. There will be four engineers working on the software, they will be responsible for frontend development, backend development, and quality assurance testing among other things. In addition, we will need to work closely with our client to provide us some insight into the business aspect of things as well as what they expect from us. All engineers are required to work a standard 40-hour work week, for about 10 weeks. Engineering estimates this software to be completed in the next two months, roughly by the beginning of August 2024. There are five key milestones to be kept track of, along with their **estimated dates of completion**, these include:

- Gathering information and writing a software requirements specification document. **2024-06-7**
- Writing a user-manual for the software. **2024-06-19**
- Designing the software interface (software design specifications). **2024-07-03**
- Writing and compiling the software. **2024-07-17**
- Software linking, testing, and debugging. **2024-07-31**

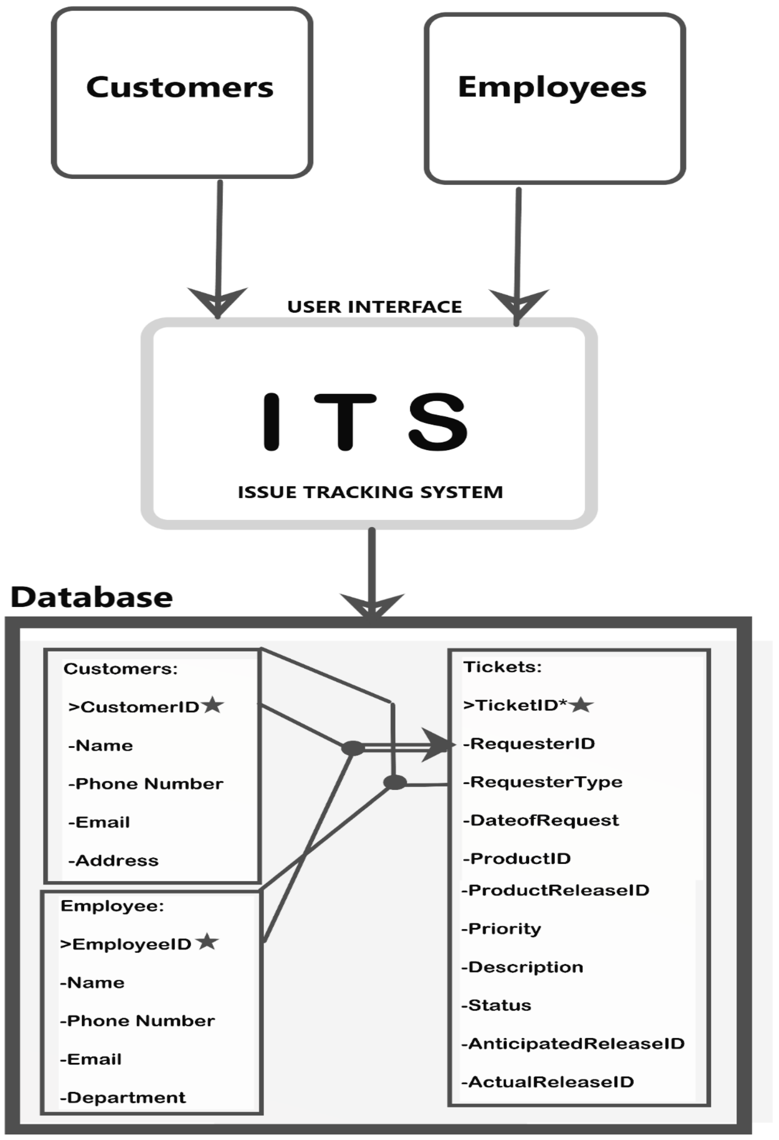
The most suitable development life cycle for this software project is the Agile development life cycle as it allows us to break down our software development into the key milestones listed above, thus giving the client frequent deliverables on the status of our software. This software will be developed in the C++ language and will be tested on the Windows OS.

2.0 Environment

2.1 Context DFD Diagram and Brief Overview of Interfaces to Other Systems

Context DFD Diagram

The Level 0 Data Flow Diagram (DFD) provides an overview of the main processes and data flows within the Issue Tracking System (ITS). It illustrates the relationship of the ITS with external entities such as customers, employees, and the database



Description of the Diagram:

1. Customers (External Entity)
   - Input: Bug reports, feature requests, general inquiries.
   - Output: Confirmation of issue submission, status updates.
2. Employees (External Entity)
   - Input: Internal change requests, status updates for issues.
   - Output: Reports, notifications of issue status changes.
3. Database (Internal Entity)
   - Input: All data related to customers, employees, issues, and product releases.
   - Output: Data retrieval for reports and status updates.
4. Tickets(Internal Process)
   - Input: Requests for tickets from employees or management.
   - Output: Generated tickets on issue statuses, customer interactions, and other analytics.

Data Flows:

- Customer to ITS:
  - Customers provide their details and issue descriptions to an employee over the phone.
  - An Employee files a report on their behalf
  - The system validates and stores this information in the database.
  - The system sends confirmation messages back to customers.
- Employee to ITS:
  - Employees enter change requests and update issue statuses.
  - The system updates the database accordingly.
  - The system generates reports and sends notifications back to employees.
- ITS to Database:
  - The system performs CRUD (Create, Read, Update, Delete) operations on the database.
  - Data such as customer information, issue details, and product releases are stored and retrieved as needed.
- ITS to Tickets:
  - The system Identifies the reports with a generated unique ticket number.
  - The system processes requests for tickets based on the data in the database.
  - The system generates and formats the tickets for viewing and printing.

Boundary Discussion:

Networks:

- Connection Type: Ethernet.(or OS supported secure network type)
- Protocols: TCP/IP.
- Purpose: Enables communication between the ITS, external users (customers, employees), and remote servers for data backup and updates.

Files:

- Types: Spreadsheets .xlsx or .csv (e.g., Microsoft Excel), text files, and reports.
- Purpose: Allows data export for analysis and reporting, as well as data import for system updates.
- Integration: The ITS must be able to read and write these files to facilitate data exchange with other systems.

Different Classes of Users:

- Customers:
    - Role: Report bugs, request features, inquire about issue statuses.
- Employees:
    - Roles:
        - Customer Support Staff: Handle customer inquiries and issue reports.
        - Marketing/Sales Staff: Submit feature requests based on customer feedback.
        - Software Engineers/QA Testers: Review, update, and resolve issues.
        - Management: Review reports and monitor system performance.
    - Interface: GUI for entering data, updating issue statuses, and generating reports.

Printers:

- Types: Standard office printers.
- Purpose: Print reports and tickets.
- Requirements: Must support various print speeds (e.g., 20 ppm or higher) and resolutions (e.g., 300 dpi or higher), with optional color support.

Discussion:

- The ITS needs to seamlessly integrate with the corporate network infrastructure, ensuring secure and reliable communication. It must support file operations to enable data exchange with external systems, particularly for reporting and updates. The system's GUI must cater to different user roles, providing intuitive interfaces for employees. The printing functionality must be compatible with standard office printers to facilitate the production of hardcopy reports and tickets. The integration of these components ensures that the ITS operates efficiently within the existing IT ecosystem of the organization.

2.2 Hardware

Recommended Hardware Specifications:

To ensure the Issue Tracking System (ITS) operates efficiently and meets the needs of users, the following hardware specifications are recommended. These specifications consider both performance and capacity requirements to handle the expected load of the system.

Minimum Hardware Requirements:

1. Processor:
    - Type: Dual-core processor
    - Performance: At least 25 MIPS (Million Instructions Per Second)
    - Examples: Intel Core i3, AMD Ryzen 3, or equivalent
    - Reason: A dual-core processor provides sufficient computational power for running the ITS smoothly, handling multiple tasks, and ensuring quick response times for users.
2. RAM:
    - Capacity: Minimum 8 GB
    - Reason: Adequate RAM ensures that the system can handle multiple concurrent users and processes without significant slowdowns, providing a smooth user experience.
3. Storage:
    - Capacity: At least 100 GB of available disk space
    - Type: SSD (Solid State Drive) recommended for faster read/write speeds
    - Reason: Sufficient storage is required to maintain the database, logs, and other necessary files. An SSD enhances the overall performance of the system, reducing load times and improving data access speeds.
4. Network:
    - Type: Ethernet connection with standard TCP/IP support
    - Reason: A reliable Ethernet connection ensures stable and fast communication between the ITS and external entities, such as customers and employees, as well as remote servers for backups and updates.
5. Printers:
    - Type: Standard office printers
    - Specifications:
        - Print Speed: 20 pages per minute (ppm) or higher
        - Resolution: 300 dots per inch (dpi) or higher
        - Color Support: Optional, but beneficial for printing detailed reports and graphs
    - Reason: The ITS needs to support common office printers to facilitate the printing of reports and tickets. The specified print speed and resolution ensure that printed documents are produced quickly and with high quality.

Recommended Hardware Configuration for Optimal Performance:

1. Processor:
   - Type: Quad-core processor
   - Performance: At least 50 MIPS
   - Examples: Intel Core i5 or i7, AMD Ryzen 5 or 7, or equivalent
   - Reason: A quad-core processor provides enhanced performance for handling more intensive tasks, running complex queries, and supporting a larger number of concurrent users without performance degradation.
2. RAM:
   - Capacity: 16 GB or higher
   - Reason: More RAM allows for better multitasking and ensures that the system can handle more simultaneous users and larger datasets without performance issues.
3. Storage:
   - Capacity: 250 GB or higher
   - Type: SSD
   - Reason: Increased storage capacity accommodates larger databases and more extensive logging, while an SSD ensures fast data access and reduced downtime.
4. Network:
   - Type: Gigabit Ethernet connection
   - Reason: Gigabit Ethernet provides faster data transfer rates, essential for environments with high data traffic and remote access requirements.
5. Printers:
   - Type: High-speed office printers
   - Specifications:
     - Print Speed: 30 pages per minute (ppm) or higher
     - Resolution: 600 dots per inch (dpi) or higher
     - Color Support: Recommended for detailed reports and presentations
   - Reason: High-speed printers with better resolution improve productivity and the quality of printed materials, especially for detailed reports and presentations.

Hardware Justification:

The recommended hardware configuration ensures that the ITS operates efficiently and effectively within a typical corporate IT environment. By specifying a minimum and optimal hardware setup, the system can be scaled according to the organization's size and requirements. These specifications provide a balance between performance and cost, ensuring that the ITS can handle the expected workload and deliver a smooth user experience.

2.3 Operating System

Operating System Requirements:

To ensure optimal performance and compatibility, the Issue Tracking System (ITS) must run on specific operating systems. This section outlines the recommended operating system requirements, including versions and justifications for these choices.

Recommended Operating System:

1. Compatibility:
- Operating System: Windows 10 or later
- Reason: Windows 10 is widely adopted in corporate environments, offering robust security features, stability, and comprehensive support for business applications. It provides a familiar user interface, reducing the learning curve for users.

2. Detailed Specifications:

Windows 10:
- Version: Windows 10 (version 1909 or later)
- Why Windows 10?
  - Security: Windows 10 includes advanced security features such as Windows Defender, BitLocker, and Windows Information Protection. These features help protect the ITS and its data from malware, unauthorized access, and other security threats.
  - Stability: Windows 10 is known for its stability and reliability, which are crucial for maintaining consistent uptime and performance of the ITS.
  - Support: Microsoft provides regular updates and extended support for Windows 10, ensuring that the operating system remains secure and up-to-date with the latest features and improvements.
  - Compatibility: Windows 10 supports a wide range of hardware and software, ensuring compatibility with the various components and peripherals used in the ITS.

Windows 11:
- Version: Windows 11 (any release)
- Why Windows 11?
  - Enhanced Features: Windows 11 offers improved performance, enhanced security features, and a modernized user interface, providing an even better user experience.
  - Future-proofing: Adopting Windows 11 ensures compatibility with future updates and software enhancements, providing a long-term solution for the ITS.

    - Compatibility: Similar to Windows 10, Windows 11 supports a wide range of hardware and software, ensuring seamless integration with the ITS components.

Additional Considerations:

1. Network Requirements:
    - Network Protocol: TCP/IP
    - Network Configuration: The ITS should be configured to use a reliable Ethernet connection to ensure stable and fast communication with external entities such as employees, and remote servers.
2. User Accounts and Permissions:
    - User Accounts: The operating system must support the creation of multiple user accounts with varying levels of access and permissions to ensure security and proper data access control.
    - Permissions: Administrators should have full control over the system, while regular users (employees) should have restricted access based on their roles and responsibilities.
3. Backup and Recovery:
    - Backup Solutions: The operating system should support backup solutions that allow for regular backups of the ITS data. This ensures data integrity and quick recovery in case of system failures or data corruption.
    - Recovery Tools: The OS should include recovery tools to restore the system to a previous state, minimizing downtime and data loss.

Justification for Choosing Windows 10 or 11:

1. Widespread Use:
    - Windows 10 and 11 are widely used in corporate environments, making them a standard choice for business applications. This widespread adoption ensures that users are familiar with the OS, reducing training costs and improving productivity.
2. Advanced Security Features:
    - Both Windows 10 and 11 offer advanced security features that protect the system and data from various threats. These features are crucial for maintaining the confidentiality, integrity, and availability of the ITS.
3. Regular Updates and Support:
    - Microsoft provides regular updates and extended support for both Windows 10 and 11, ensuring that the operating system remains secure and up-to-date with the latest features and improvements.
4. Compatibility and Integration:
    - Windows 10 and 11 are compatible with a wide range of hardware and software, ensuring seamless integration with the ITS components. This compatibility reduces the risk of technical issues and simplifies system maintenance.

By specifying Windows 10 or later as the required operating system for the ITS, we ensure that the system operates on a secure, stable, and widely supported platform. This choice provides a robust foundation for the ITS, enabling it to meet the needs of customers and employees effectively.

2.4 Database

Database Management System (DBMS) Requirements:

The Issue Tracking System (ITS) relies on a robust and scalable **Custom** database management system to store, retrieve, and manage the data associated with customer reports, employee records, issue logs, and product release information. This section outlines the recommended DBMS, the rationale for its selection, and the specific requirements and considerations for its implementation.

Database Schema and Design:

1. Schema Definition:
   - Tables:
     - Customers: Stores customer information.
     - Employees: Stores employee information.
     - Issues: Stores details of reported issues and feature requests.
     - ProductReleases: Stores information about product releases.
2. Relationships:
   - Foreign Keys: Link issues to customers, employees, and product releases to ensure data integrity and consistency.
   - Normalization: The database should be normalized to at least the third normal form (3NF) to reduce data redundancy and improve data integrity.
3. Sample Data:
   - Customers Table:
     - CustomerID, Name, Phone, Email
   - Employees Table:
     - EmployeeID, Name, Phone, Email, Department
   - Issues Table:
     - IssueID, RequesterID, RequesterType, DateOfRequest, ReleaseID, Priority, Description, Status, AnticipatedReleaseID, ActualReleaseID
   - ProductReleases Table:
     - ReleaseID, ProductID, ProductName, ReleaseNumber, ReleaseDate

Implementation Considerations:

1. Data Security:

- Access Control: Implement role-based access control (RBAC) to restrict access to sensitive data. Only authorized personnel should be able to view or modify specific data.
- Encryption: Use encryption for data at rest and in transit to protect sensitive information from unauthorized access and breaches.

2. Backup and Recovery:
   - Regular Backups: Schedule regular backups of the database to prevent data loss in case of hardware failures, software issues, or other unexpected events.
   - Recovery Procedures: Establish and document recovery procedures to restore the database quickly and efficiently in case of data corruption or loss.
3. Performance Optimization:
   - Indexing: Use indexing to improve the performance of database queries, especially for frequently accessed tables and columns.
   - Query Optimization: Optimize SQL queries to reduce the load on the database and improve response times.
4. Maintenance:
   - Monitoring: Implement monitoring tools to track the performance and health of the database, including metrics such as query response times, transaction rates, and resource usage.
   - Updates: Keep the DBMS updated with the latest patches and updates to ensure security and performance enhancements.

2.5 Development Environment

Development Environment Requirements:

The development environment for the Issue Tracking System (ITS) must provide the necessary tools and resources to support efficient coding, debugging, testing, and project management. This section outlines the recommended Integrated Development Environment (IDE), software development kits (SDKs), compilers, and other tools required for developing the ITS using C++.

Recommended Development Environment:

1. Integrated Development Environment (IDE):
   - IDE: Visual Studio 2019 or later
   - Reason: Visual Studio provides a comprehensive development environment with integrated tools for coding, debugging, and project management. It supports multiple programming languages, including C++, which is used for the ITS.
2. Software Development Kits (SDKs):

- SDK: Windows SDK
- Reason: The Windows SDK provides the necessary libraries and tools for building Windows applications in C++. It integrates seamlessly with Visual Studio and supports a wide range of functionalities required for the ITS.

3. Compiler:
   - Compiler: Microsoft Visual C++ Compiler
   - Reason: The Visual C++ compiler is a high-performance compiler that supports the latest C++ standards and provides extensive error checking and optimization features.
4. Debugger:
   - Debugger: Integrated debugger in Visual Studio
   - Reason: Visual Studio's debugger offers powerful debugging features, including breakpoints, watch variables, and step-through execution, which are essential for identifying and fixing bugs during development.
5. Version Control:
   - System: Git
   - Hosting Service: GitHub, GitLab, or Azure DevOps
   - Reason: Version control is critical for managing code changes, collaborating with team members, and maintaining a history of project development. Git is widely used and supported by various hosting services, providing robust version control capabilities.
6. Testing Framework:
   - Framework: Google Test (gtest) or Catch2
   - Reason: These testing frameworks provide comprehensive support for unit testing in C++ applications, enabling developers to write and run tests to ensure the correctness and reliability of the code.
7. Additional Tools:
   - Code Analysis: ReSharper C++ (optional)
     - Reason: ReSharper C++ enhances the coding experience by providing advanced code analysis, refactoring, and navigation features.
   - Build Automation: Azure Pipelines or GitHub Actions
     - Reason: These tools automate the build and deployment process, ensuring continuous integration and delivery. This helps maintain code quality and accelerates the development lifecycle.

Detailed Specifications:

Visual Studio 2019 or Later:

- Features:
  - Comprehensive code editor with IntelliSense for C++
  - Integrated debugging and profiling tools
  - Project templates and wizards for C++ development
  - Built-in support for version control systems

    - Extensions for enhanced functionality (e.g., ReSharper C++)

Windows SDK:

- Features:
    - Header files, libraries, and tools for Windows development
    - Documentation and sample code for Windows APIs
    - Integration with Visual Studio for seamless development

Microsoft Visual C++ Compiler:

- Features:
    - Support for the latest C++ standards (C++11, C++14, C++17, or newer)
    - Advanced optimization and code generation techniques
    - Extensive error checking and diagnostics

Google Test (gtest) or Catch2:

- Features:
    - Comprehensive framework for writing and running unit tests
    - Support for test fixtures, assertions, and test discovery
    - Integration with build systems and IDEs

Azure Pipelines or GitHub Actions:

- Features:
    - Continuous integration and continuous delivery (CI/CD) pipelines
    - Automated build and test processes
    - Integration with version control systems
    - Support for various build environments and configurations

Implementation Considerations:

1. Code Quality:
    - Code Reviews: Regular code reviews to ensure code quality and adherence to coding standards.
    - Static Analysis: Use static analysis tools to detect potential issues and improve code quality.
2. Collaboration:
    - Communication Tools: Use tools like Slack or Microsoft Teams for team communication and collaboration.
    - Documentation: Maintain comprehensive documentation for code, APIs, and development processes.
3. Continuous Integration:
    - Automated Builds: Set up automated build processes to compile and test code changes.
    - Automated Tests: Integrate unit tests into the CI pipeline to ensure code correctness.
4. Security:
    - Secure Coding Practices: Follow secure coding practices to prevent vulnerabilities.

- Regular Updates: Keep development tools and libraries updated to the latest versions to avoid security risks.

By specifying Visual Studio 2019 or later as the IDE, the Microsoft Visual C++ Compiler, and additional tools like Google Test or Catch2 for testing, the ITS development environment provides a robust and comprehensive setup for developing a high-quality issue tracking system in C++. This environment ensures efficient development, testing, and maintenance of the ITS, enabling the team to deliver a reliable and scalable solution.

2.6 Maintenance Environment

Maintenance Environment Requirements:

The maintenance environment for the Issue Tracking System (ITS) ensures the system remains functional, up-to-date, and secure over its operational life. This section outlines the recommended maintenance practices, responsibilities, tools, and procedures necessary for effective system maintenance.

1. Responsibility:
Internal IT Department:

- Role: The internal IT department will be responsible for the maintenance of the ITS.
- Tasks: Regular updates, troubleshooting issues, ensuring system security, and implementing enhancements.

Reason: The internal IT department has the expertise and resources required to manage the ITS effectively, ensuring its continuous operation and addressing any issues that arise promptly.
2. Deployment of Enhancements:
Scheduled Maintenance Windows:

- Reason: Enhancements and updates will be deployed during scheduled maintenance windows to minimize disruption to users. This approach ensures that updates are applied without affecting system availability during critical business hours.

3. Data Migration:
Automated Scripts:

- Role: Automated scripts will facilitate data migration during system updates or when transitioning to new versions.
- Reason: These scripts ensure accurate and efficient data transfer, minimizing the risk of data loss or corruption.

Backup Procedures:

- Regular Backups: Schedule regular backups of the database to prevent data loss.
- Backup Tools: Use reliable backup tools that support incremental and full backups.

- Storage: Store backups securely, both onsite and offsite, to ensure data availability in case of a disaster.

4. Update Transmission:
Methods:
- Secure FTP: Use secure FTP for transmitting updates.
- Cloud Storage: Utilize cloud storage solutions for distributing updates.
- Reason: These methods ensure safe and efficient delivery of updates to all users.

5. Remote Diagnosis:
Remote Access Tools:
- VPN Connections: Secure VPN connections for remote access.
- Remote Monitoring Tools: Tools like TeamViewer or Remote Desktop for remote troubleshooting.
- Reason: Remote support allows IT staff to diagnose and resolve issues quickly, even if they are not physically present at the location where the ITS is installed.

6. Monitoring and Logging:
Monitoring Tools:
- Tools: Use monitoring tools like Nagios, Zabbix, or SolarWinds to track the performance and health of the ITS.
- Metrics: Monitor key metrics such as server uptime, database performance, and network latency.

Logging:
- Error Logs: Maintain detailed error logs to track and troubleshoot issues.
- Access Logs: Keep access logs to monitor user activity and ensure security.

7. Security:
Access Control:
- Role-Based Access Control (RBAC): Implement RBAC to restrict access to sensitive data.
- Reason: Ensures that only authorized personnel can view or modify specific data, enhancing security.

Encryption:
- Data at Rest: Encrypt sensitive data stored in the database.
- Data in Transit: Use SSL/TLS for encrypting data transmitted over the network.
- Reason: Encryption protects sensitive information from unauthorized access and breaches.

8. Maintenance Procedures:
Documentation:
- Maintenance Plan: Develop a comprehensive maintenance plan detailing regular maintenance tasks, schedules, and responsibilities.
- Change Management: Document all changes made to the system, including updates, and configuration changes.

Regular Maintenance Tasks:

- System Updates: Regularly update the operating system, database management system, and application components to the latest versions.
- Security Patches: Apply security patches promptly to address vulnerabilities.
- Performance Tuning: Regularly monitor and optimize system performance.

Incident Management:

- Incident Response Plan: Develop and maintain an incident response plan to handle system outages, security breaches, and other emergencies.
- Training: Train IT staff on incident response procedures to ensure quick and effective resolution of issues.

9. Communication:

User Notifications:

- Announcements: Inform users about scheduled maintenance, updates, and any potential downtime.
- Support Channels: Provide multiple support channels (e.g., email, phone, helpdesk system) for users to report issues and request assistance.

Feedback Mechanism:

- User Feedback: Establish a feedback mechanism to gather user input on system performance and areas for improvement.
- Continuous Improvement: Use feedback to continuously improve the system and address user needs.

Patching:

- Note: Patching individual features is not required for the ITS.
- Reason: The ITS is designed to handle updates and enhancements without the need for frequent patches. Full updates are applied during scheduled maintenance windows.

Conclusion:

The maintenance environment for the ITS is designed to ensure the system remains functional, secure, and up-to-date throughout its lifecycle. By following these guidelines and procedures, the internal IT department can effectively manage the ITS, ensuring its continuous operation and addressing any issues promptly. This proactive approach to maintenance helps minimize downtime, enhance system performance, and maintain user satisfaction.

3.0: User Interface and Operations

3.1: User Interface Sophistication

The user interface of the Issue Tracking System (ITS) is designed to provide a seamless and efficient experience for support staff and engineers. Users interact with the system through a series of well-organized menus and forms by selecting a menu option they would like to proceed with. The main menu gives the user options to quit the program or select if he/she is an employee in order to proceed to the proper data collection and submenus. A submenu provides quick access to core functionalities such as reporting a bug, requesting a feature, viewing the status of submitted issues, updating a ticket and its attributes, and quitting. Each function is accompanied by clear prompts and guidance, reducing the likelihood of errors. Error messages and guidance prompts are displayed in real-time to assist users in correcting any mistakes quickly. The system is designed to handle all interactions on a single screen with no need for scrolling, ensuring that users can complete their tasks efficiently.

3.2: Use Case Scenarios

Use Case 1:

Title: Reporting a Bug/Requesting a Feature

Actor: As a customer support staff member for a customer

Preconditions:

- The issue tracking system is open and running.
- The customer support staff member is logged into the system.

Steps of the Interaction:

1. Prompt asks the user if it is for a customer or a staff member.
2. User selects customer.
3. System prompts for email address.
4. User enters the customer’s email.
5. System checks the database to see if the customer is already registered.
6. If the customer is new, the system prompts for the customer's name, phone number, and email address.
7. User enters the customer’s information.
8. System saves the customer's information.
9. System prompts a menu of actions (create ticket, check updates, inquire, quit).
10. User clicks on "Create New Ticket".

11. System prompts for the product release ID.
12. User enters the product release ID.
13. System checks for the validity of the product release ID.
14. If valid, the system prompts for the issue details (type, title, description, priority).
15. User enters the ticket details.
16. System assigns a unique ID to the new ticket.
17. System displays a confirmation message.

Normal Results:
The ticket is successfully logged into the system and assigned a unique ID.

Other Variants:
- If the product release ID is invalid, the system displays an error message and prompts the user to re-enter the correct ID.
- If the user is already registered, the system skips steps 6 to 8.

Use Case 2:

Title: Reporting a Bug/Requesting a Feature by employee

Actor: As a customer support staff member for another department staff member

Preconditions:
- The issue tracking system is open and running.
- The customer support staff member is logged into the system.

Steps of the Interaction:
1. Prompt asks the user if it is for a customer or a staff member.
2. User selects staff member.
3. System prompts for employee’s ID.
4. User enters the employee’s ID.
5. System checks the database to see if the employee is already registered.
6. If the staff member is new, the system prompts for the customer's name, phone number, department, and email address.
7. User enters the customer’s information.
8. System checks for the validity of the staff department.
9. If valid, the system saves the staff's information; otherwise, it rejects it and goes back to the main menu.
10. System prompts a menu of actions (create ticket, inquiries, request report, quit).
11. User clicks on "Create New Ticket".
12. System prompts for the product release ID.

13. User enters the product release ID.
14. System checks for the validity of the product release ID.
15. If valid, the system prompts for the issue details (type, title, description, priority).
16. User enters the ticket details.
17. System assigns a unique ID to the new ticket.
18. System displays a confirmation message.

Normal Results:
The ticket is successfully logged into the system and assigned a unique ID.

Other Variants:
- If the product release ID is invalid, the system displays an error message and prompts the user to re-enter the correct ID.
- If the staff is already registered, the system skips steps 6 to 8.

Use Case 3:

Title: Updating Bug/Enhancement Status

Actor: As a software engineer

Preconditions:
- The issue tracking system is open and running.
- The software engineer is logged into the system.
- The issue to be updated exists in the system.

Steps of the Interaction:
1. Prompt asks the user if it is for a customer or staff member.
2. User selects a staff member.
3. System prompts for employee's ID.
4. User enters the employee's ID.
5. System prompts a menu of actions (create ticket, update a ticket, request report, edit product release records, quit); otherwise, it rejects it and goes back to the main menu.
6. User clicks on update a ticket.
7. Systems prompts a menu of actions (search by ticket ID, search by release ID, quit) .
8. User clicks on search by ticket ID.
9. System prompts for ticket ID.
10. User enters a ticket ID.
11. System displays the ticket details.
12. User clicks on "Update Status".
13. System prompts for the new status (e.g., In Progress, Done, Canceled).

14. User selects the new status and enters any additional comments.
15. System saves the updated status and comments.
16. System displays a confirmation message.

Normal Results:
The issue status is updated and saved in the system.

Other Variants:
- If the employee ID is invalid, the system displays an error message and prompts the user to re-enter the correct ID.
- If the ticket ID is invalid, the system displays an error message and prompts the user to re-enter the correct ID.
- If the status is set to "Done", the system prompts for the actual release ID where the fix is implemented.

Use Case 4:

Title: Inquiring About a Particular Change

Actor: As a customer support staff member for a customer

Preconditions:
- The issue tracking system is open and running.
- The user has logged in through the system prompts.
- The change item to be inquired about exists in the system.
- The user is already registered.

Steps of the Interaction:
1. System prompts the user to select if it is for a customer or staff member.
2. User selects customer.
3. System prompts for email address.
4. User enters the customer's email address.
5. System prompts a menu of actions (create ticket, check updates, edit product release records, request report, quit).
6. User selects the option inquiries.
7. System prompts the user to select the method for inquiring (by ticket ID, product ID release, or list).
8. User selects the customer's preferred method (e.g., entering the change ID).
9. System retrieves and displays the details of the selected item.

Normal Results:

- The details of the selected item ticket are displayed to the user, including ticket ID, product, description, priority, status, anticipated release, and actual release.

Other Variants:
- If the change ID is invalid, the system displays an error message and prompts the user to re-enter the correct ID.

Use Case 5:

Title: Adding and Deleting Product Release Records

Actor: As software engineer staff

Preconditions:
- The issue tracking system is open and running.
- The software engineer has logged in through the system prompts.
- The user is already registered.

Steps of the Interaction:
Adding a Product Release Record:
1. System prompts the user to select if it is for a customer or staff member.
2. User selects "staff member".
3. System prompts for employee's ID.
4. User enters the employee's ID.
5. System prompts a menu of actions (create ticket, check updates, edit product release records, request report, quit).
6. User selects "edit product release records".
7. Systems prompts a three option menu: add product release record, delete product release record, quit.
8. User selects the option to add a product release record.
9. System prompts for the product release details (e.g., product name, release ID, release date).
10. User enters the product release details.
11. System saves the new product release record.
12. System displays a confirmation message.

Deleting a Product Release Record:
1. System prompts the user to select if it is for a customer or staff member.
2. User selects "staff member".
3. System prompts for employee's ID.
4. User enters the employee's ID.

5. System prompts a menu of actions (create ticket, check updates, edit product release records, request report, quit).
6. User selects "edit product release records".
7. User selects the option to delete a product release record.
8. System prompts for the product release ID to be deleted.
9. User enters the product release ID.
10. System validates the product release ID.
11. If valid, system deletes the product release record.
12. System displays a confirmation message.

Normal Results:

- Product release records are added or deleted as needed.

Other Variants:

- If the product release ID is invalid, the system displays an error message and prompts the admin to re-enter the correct ID.

Use Case 6:

Title: Displaying Newly-Reported Change Items

Actor: As a software engineer

Preconditions:

- The issue tracking system is open and running.
- The software engineer has logged in the system.
- There are newly-reported change items in the system.
- The employee is already registered.

Steps of the Interaction:

1. System prompts the user to select if it is for a customer or staff member.
2. User selects "staff member".
3. System prompts for employee’s ID.
4. User enters the employee’s ID.
5. System prompts a menu of actions (create ticket, check updates, edit product release records, request report, quit).
6. User selects "check updates".
7. System displays a list of newly-reported change items.
8. Software engineer reviews each change item.
9. For each change item, the software engineer assesses its validity.

10. If the change item is invalid or redundant, the engineer changes the status to "Canceled" and amends the description.
11. If the change item is valid, the engineer assigns an anticipated release ID for the fix and possibly changes the priority.
12. System saves the updated information for each change item.

Normal Results:

- Newly-reported change items are assessed, and their statuses, priorities, and anticipated release IDs are updated accordingly.

Other Variants:

- If there are no newly-reported change items, the system displays a message indicating that there are no new items to review.

Use Case 7:

Title: Generating a Report of Changes for a Product

Actor: As a manager

Preconditions:

- The issue tracking system is open and running.
- The manager has logged in through the system prompts.
- There are change items related to the selected product.

Steps of the Interaction:

1. System prompts the user to select if it is for a customer or staff member.
2. User selects "staff member".
3. System prompts for employee’s ID.
4. User enters the employee’s ID.
5. System prompts a menu of actions (create ticket, check updates, request report, quit).
6. User selects "request report".
7. System displays report generation options.
8. Manager selects the option to generate a report of changes for a particular product.
9. System prompts for the product and additional criteria (e.g., status: not implemented, anticipated release).
10. Manager enters the product and criteria.
11. System generates the report based on the selected criteria.
12. System displays the report and provides options to export it in various formats (e.g., PDF, Excel).

Normal Results:

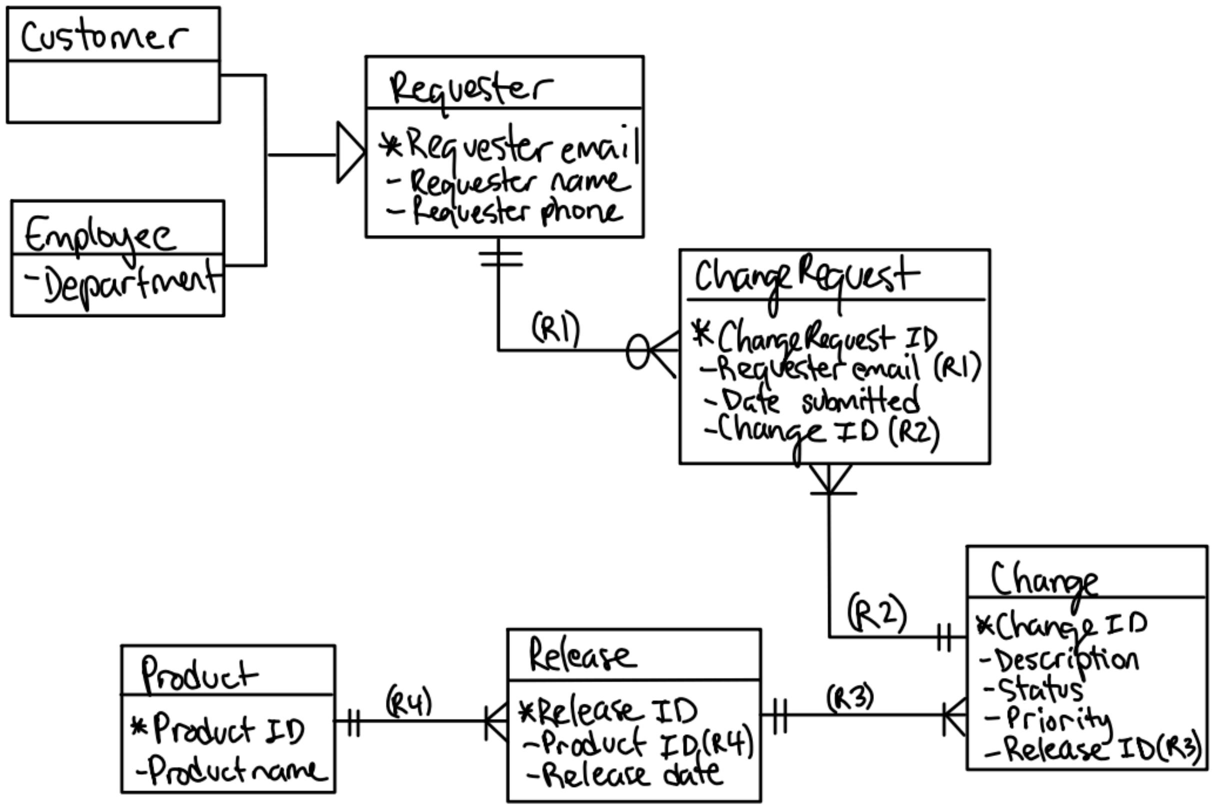
- The report of changes for the selected product is generated and displayed, and can be exported as needed.

Other Variants:

- If no changes match the selected criteria, the system displays a message indicating no results were found.

4.0: Retained Data Model

4.1: Object Relationship Diagram



4.2: Discussion and Justification of Object Relationship Diagram

4.2.1: Cardinality

1. **Relationship R1 — Requester and Change Request**
   - A requester has zero or more change requests
     - A requester (customer or employee) may submit multiple change requests over time, or can have no change requests at any given time, such as after their requests have been addressed.
     - The optionality on the right side cannot be one because a requester can exist without having any current change requests.

        - The multiplicity on the right side cannot be one because a single requester can submit multiple change requests.

    - Each change request belongs to one and only one requester
        - Each change request is submitted by one specific requester.
        - The optionality on the left side cannot be zero because a change request cannot exist without being submitted by a requester.
        - The multiplicity on the left side cannot be many because a single change request cannot belong to multiple requesters.

2. **Relationship R2 — Change Request and Change**
    - A change request belongs to one and only one change
        - Each change request submission requests for a single change.
        - The optionality of the right side cannot be zero because there cannot be a change request that does not ask for a change.
        - The multiplicity of the right side cannot be many because a change request cannot request multiple changes, users would have to submit multiple requests.
    - A change has one or more change requests
        - A single change can be requested many times; a change might have been identified independently by several requesters.
        - The optionality of the left side cannot be zero because there would not be a change made if there were no requests.
        - The multiplicity of the left side cannot be one because there can be many change requests.

3. **Relationship R3 — Release and Change**
    - A release has one or more changes
        - Every new release of a product has one or more changes, reflecting the new features or bug fixes.
        - The optionality on the right side cannot be zero; a release cannot exist without any changes included.
        - The multiplicity on the right side cannot be one because a new release can have multiple changes and is not limited to just one.
    - Each change belongs to one and only one specific release
        - Each change is scheduled for a specific release.
        - The optionality on the left side cannot be zero because a change must be part of a release.
        - The multiplicity on the left side cannot be many because one specific change is typically associated with one release.

4. **Relationship R4 — Product and Release**
    - A product can have one or more releases
        - A product undergoes multiple releases, each representing a new version with enhancements or fixes.
        - The optionality on the right side cannot be zero because there cannot be a product with no releases.
        - The multiplicity on the right side cannot be one because a product can have multiple releases and is not limited to just one.
    - A release belongs to one and only one product
        - Each release is product specific and correlates to one product indicating that the changes and updates pertain to that particular product.
        - The optionality on the left side cannot be zero because a release must correspond to a product.
        - The multiplicity on the left side cannot be many because a single release cannot be associated with multiple products.

4.2.2: Key Uniqueness

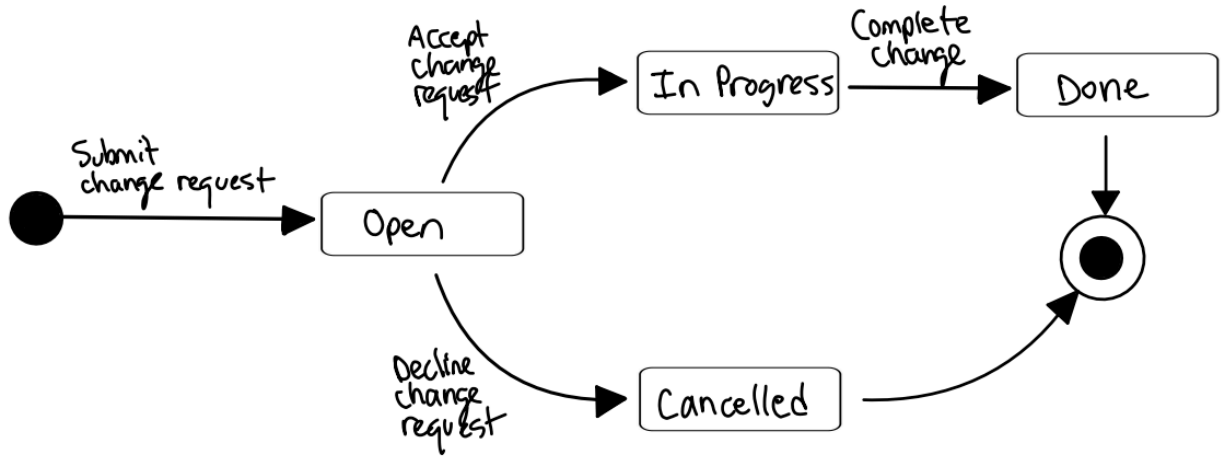
The requester's email, the change request ID, the change ID, the product ID, and release ID, are all unique because they are used as the primary keys of the entities in the ORD. The email for each requester serves as an unique identifier as customers and/or employees may have duplicate names. The change request ID allows for precise tracking of each request from submission. The change ID ensures each implemented change can be distinctly identified and associated with specific requests and releases. The product ID uniquely identifies each product, allowing for accurate management of changes and releases. Finally, the release ID uniquely identifies each product release, facilitating clear version control and documentation. These unique primary keys are crucial for maintaining data integrity, accurate tracking, and efficient management within the system.

4.2.3: Data Exception Handling

The requester must submit a valid email address or phone number. New requesters must also fill in all required fields (name, email, phone). The system will validate email and phone number formats, as well as check for the presence of all required fields, otherwise the system will display an error message to the user and prevent the change request from being submitted. To mitigate input of incorrect product names or non-existent products, requesters will select from a list of products.

5.0: State (Control) Modes

5.1: State Diagram



5.2: State Discussion

The lifecycle of a change request in the issue tracking system is shown through a state diagram consisting of four states: Open, In Progress, Cancelled, and Done. The process begins when a requester submits a change request.

When a user submits a change request, the state of the request is automatically Open. Upon reviewing a change request, if it is declined, the state becomes Cancelled. This may occur if the request does not make sense, has already been handled in a different change request, or if it is determined upon further investigation to just be a user problem. If a change request is accepted, the change process will begin and the state of the request transition to be In Progress.

Upon completion of the change, the state of the request will transition to Done, signifying that the requested change has been successfully implemented and finalized. The state of the request cannot be changed from the Cancelled or Done state. This structure ensures that every change request is systematically processed, either through completion or cancellation, and appropriately finalized.

6.0: Dictionary

Composite terms:

Letter = “A” | “B” | “C” | … | “X” | “Y” | “Z” | “a” | “b” | “c” | … | “x” | “y” | “z”
Digit = “0” | “1” | “2” | "3" | "4" | "5" | "6" | "7" | "8" | "9"
SpecialChar = "!" | "#" | "$" | "%" | "&" | "'" | "*" | "+" | "-" | "/" | "=" | "?" | "^" |
"_" | "`" | "{" | "|" | "}" | "~"
Character = Letter | Digit | SpecialChar
LocalPart = Character { Character }
Label = Letter { Letter | Digit } Letter
Domain = Label { "." Label }

Accept — The action of approving a change request

Cancelled — A state indicating that the change request has been denied

Change — An object that includes change ID, description, priority, status and release ID. It is an alteration, addition, or bug fix of a product.

Change ID — A unique identifier for a change that is a fixed length string used as the primary key for the change entity with the format “CHG”{digit}^{5}

Change Request — An object that includes change request ID, requester information, submission date, change ID, and is a formal request for a change to be made to a product

Change Request ID — A unique identifier for a change request that is a fixed length string with the format “CRQ” {digit}^{5}

Complete — The action of finishing a change

Customer — A subclass of requester who is external to the organization

Decline — The action of rejecting a change request

Department — The department within the organization that the employee requester belongs to

Description — A detailed description of the change being requested that is a string of variable length with the maximum length of 300 characters

Done — A state indicating that the change has been completed and implemented

Employee — A subclass of requester who is an internal member of the organization

In Progress — A state indicating that the change is currently being worked on

Open — A state indicating that a change request has been submitted

Priority — The level of importance assigned to a change request or change (e.g., low, medium, high)

Product — An object including product ID and product name, and is an item that is subject to changes and change requests

Product ID — A unique identifier for a product that is a fixed length string with the format $\{letter\}^3\{digit\}^3$

Product Name — The name of the product

Release — An object that includes release ID, release date, and product ID. It is a specific version of a product that includes one or more changes

Release Date — The date on which a product release is scheduled or occurred in the format YYYY-MM-DD

Release ID — A unique identifier for a product release (e.g., 4.3.2)

Requester — An object that includes the requester's name, phone, and email. It can be a customer or employee

Requester Email — The email of the user submitting a request with the format LocalPart "@" Domain

Requester Name — The name of the user submitting a request

Requester Phone — The phone number of the user that is a fixed length string with the format +1 $\{Digit\}^3$ "-" $\{Digit\}^4$ "-" $\{Digit\}^3$

Status — The current state of the change request

Submit — The action of creating a change request

7.0: Performance

7.1 Response Times

The system's response time is a critical performance metric to ensure efficient operation. The system must meet the following response time requirements:

User Interface Operations:

- The system should respond to user actions such as form submissions and page navigations within 2 seconds under normal load conditions.
- Under high load conditions (e.g., multiple simultaneous users), the response time should not exceed 5 seconds.

Database Operations:

- Queries to the database, such as retrieving issue details, should return results within 1 second.
- Database write operations, such as creating or updating an issue, should complete within 2 seconds.

7.2 Throughput

Throughput measures the system's ability to handle a large number of transactions within a specific period. The system must be capable of the following:

Transactions per Second:

- The system should handle at least 50 transactions per second (e.g., issue updates, searches) under normal conditions.
- During peak load, the system should maintain a minimum throughput of 30 transactions per second.

Concurrent Users:

- The system should support one user at a time, but ideally it should be able to support 100 users at a time without significant performance degradation.

7.3 File Capacity

To ensure efficient data storage and management, the system should adhere to the following file capacity guidelines:

Issue Records:

- Each issue record will include fields such as title, description, status, priority, assigned employee, product release ID, and timestamps.
- The system should be capable of storing up to 100 product releases, 50 change items per release, and 2 change requests per change item.
- On average, each requester will have 1.5 requests, with 50% of requesters being internal across 15 departments.

User Records:

- The system should store details for up to 10,000 users, including identification numbers, roles, and contact information.

Audit Logs:

- Maintain audit logs for all significant actions (e.g., issue creation, updates, deletions) for at least one year.
- Logs should include details such as user ID, action performed, and timestamp.

7.4 Storage Calculation

Issue Record Size Calculation:

- Title (30 characters) + Description (100 characters) + Status (10 characters) + Priority (5 characters) + Assigned Employee (20 characters) + Product Release ID (10 characters) + Timestamps (20 characters)
- Total: 30 + 100 + 10 + 5 + 20 + 10 + 20 = 195 bytes per issue record.

User Record Size Calculation:

- Identification Number (10 characters) + Role (10 characters) + Contact Information (50 characters)
- Total: 10 + 10 + 50 = 70 bytes per user record.

Storage Capacity Calculation:

- For 100 product releases, 50 change items per release, and 2 change requests per change item: 100 * 50 * 2 = 10,000 change requests.
- Total issue records: 10,000 change requests.
- Total storage for issue records: 10,000 * 195 bytes = 1,950,000 bytes (~1.95 MB).
- Total storage for user records: 10,000 * 70 bytes = 700,000 bytes (~0.7 MB).

- Total storage for audit logs (estimated): 1,000,000 bytes (~1 MB).
- Overall total storage: 1.95 MB + 0.7 MB + 1 MB = 3.65 MB (~3.7 MB).

8.0: Software Prototype and Future Releases

8.1: Current Release

In the current release, we will include the main functionalities per the client's request. These features include:

- Report a bug
- Request a feature
- Status Update
- Exit

We know for sure that these are the main functionalities that our client wants for their software.

8.2: Prototype

The prototype will be a display menu with features that the user can choose. Features that <u>will</u> be included in the prototype are:

- Report a bug
- Request a feature
- Status Update
- Exit
- Sales inquiry
- General query
- View status of submitted issues

We are including the first four features because we know for sure they will be in the software and thus would be helpful to receive feedback about it from the client before releasing the actual software. The other three features, i.e. Sales inquiry, General query, and View status of submitted issues, are features that we think would be helpful for the software but are not completely sure about what the client thinks. Therefore, we include them in the prototype to receive clarification about them.

8.3: Future Releases

Ideas for what we could include in a future release of the software include:

- Separate tabs for viewing bugs and new features on different pages
- Separate tabs for bugs currently being worked on, bugs that have not been worked on, and bugs that are complete
- Different sorting options (i.e. sort by priority or sort by date, etc.)
- Web-based, and mobile version of the software
- Ability to have remote access of the software

9.0: Acceptance Criteria

9.1: Hard Requirements

This software must meet all requirements listed in Section 7.0 of this document.

9.2: Goals

Through our software, 5SOS aims to achieve the following goals:

- Increase productivity by at least 25%
- Receive customer and employee feedback on products more easily and in a more organized fashion

9.3: Acceptance Test Requirements

The software will go through a process of testing and debugging by our quality assurance engineers before being handed to the client. The client should run a few tests as well before determining whether the deal will go through or if there is still more testing and fixing to be done by our engineers. Once testing has concluded, the business transaction will be complete, with the client paying us the $300,000 that was agreed upon in the business contract.