Sri Lanka Institute of Information Technology

Software Engineering - SE2030

Year 2, Semester 1 – 2025

Group 8.1 - PG 145

2025-Y2-S1-MLB-B8G1-05

Web-based Customer Care System

IT Number	Name
IT24102044	Kidelpitiyage L.T
IT24102186	Jayasinghe S.S.
IT24101993	Shaheen A.N
IT24102028	Hettiarachchi D.S
IT24102137	Samarathunga S.A.J.D.M
IT24102169	Aazib A.H.M.A

B.Sc. (Hons) in Information Technology

Contents

1. **Introduction** 3
   1.1. **Project Overview** 3
   1.2. **Objectives** 3
   1.3. **Target Users or Stakeholders** 3
   1.4. **Scope and Limitations** 4
      Project Scope 4
      Limitations 4
2. **Requirements** 4
   2.1. **Functional Requirements** 4
      1) **Authentication and Core User Module** 4
      2) **Customer Module** 4
      3) **Staff Module** 4
      4) **Technician Module** 5
      5) **Product Manager Module** 5
      6) **Warehouse Manager Module** 5
      7) **Administrator Module** 5
   2.2. **Non-functional Requirements** 6
      Performance 6
      Usability 6
      Security 6
      Maintainability 6
      Reliability 6
   2.3. **Constraints or Assumptions** 7
      Technical Constraints 7
      Assumptions 7
3. **Design** 8
   3.1. **Use Case Diagram** 8
   3.2. **Database Design / ER Diagram** 9
   3.3. **UI Sketches/Screenshots and Descriptions** 10
   3.4. **System Architecture** 11
4. **Implementation** 12
   4.1. **Tools and Technologies Used** 12
   4.2. **Key Features Developed** 13
   4.3. **Screenshots of Core Functions** 14
5. **Project Management** 15

5.1. Agile Approach and Sprint Summary 15
Sprint 1 - Weeks 1-3: Foundation & Core Authentication 15
Sprint 2 - Weeks 4-6: Core Ticket Management (Customer & Staff) 15
Sprint 3 - Weeks 7-9: Repair Workflow & Technician Interface 16
Sprint 4 - Weeks 10-12: Inventory & Procurement (Product/Warehouse Manager) 16
Sprint 5 - Weeks 13-14: Admin Features & UI Refinements 17
5.2. Task Distribution Among Team Members 17
5.3. Development Phases or Timeline 18
6. Conclusion & Future Work 19
6.1. Summary of Achievements 19
6.2. Challenges Faced 19
6.3. Suggestions for Improvement or Extension 20
6.4 Conclusion 20
7. Individual Contribution, Teamwork & Lessons Learned 21
8. References 24
9. Appendix 24

1. Introduction

1.1. Project Overview

The ByteX Customer Care System is a comprehensive, role-based platform designed to streamline customer support operations, repair tracking, and inventory management. It provides a unified solution for IT hardware repair businesses to manage their entire workflow, from the initial customer support request through to repair completion, parts management, and supplier purchasing.

The application is built on a modern, layered architecture, featuring a Spring Boot backend, Spring Data JPA for database interaction, and Thymeleaf for dynamic server-side rendered web pages.

1.2. Objectives

The primary objectives of this project are:

- **To centralize operations:** Develop a single, centralized platform for managing all customer support tickets, repair jobs, and parts inventory.
- **To enforce security:** Implement strict, role-based access control (RBAC) to ensure that users can only access data and functionality relevant to their specific role.
- **To automate workflows:** Create seamless, integrated workflows between different departments, such as a technician requesting a part from the product manager, who in turn requests stock from the warehouse manager.
- **To ensure accountability:** Log all significant system activities, such as user logins or ticket status changes, for monitoring and auditing purposes.

1.3. Target Users or Stakeholders

The system is designed to serve six distinct user roles, each with a dedicated interface and set of permissions:

- **Customer:** The end-user who submits, views, and comments on their support tickets.
- **Staff:** The first line of support, responsible for managing customer communications, assigning unassigned tickets, and updating ticket statuses.
- **Technician:** The repair specialist who performs diagnoses, manages repair records, and submits part requests needed for the repair.
- **Product Manager:** Responsible for managing the parts inventory, approving part requests from technicians, and submitting stock requests to the warehouse.
- **Warehouse Manager:** Manages supplier relationships, creates and tracks purchase orders, and fulfills internal stock requests from the product manager.
- **Administrator:** A super-user with oversight of the entire system, including user management, ticket reassignment, and viewing system activity logs.

1.4.Scope and Limitations

Project Scope
This project delivers a unified web platform to manage IT repair operations through six core modules: Customer, Staff, Technician, Product Manager, Warehouse Manager, and Admin. The system's scope includes full role-based access control, user authentication with password reset, and an integrated workflow that connects a customer's support ticket to a technician's repair, a product manager's inventory, and a warehouse manager's purchase orders. The user interface is enhanced with selectable themes and a custom cursor.

Limitations
This system is a prototype and does not include production-ready security, as user passwords are not encrypted. It also lacks several features: there is no functionality for file attachments, billing, or payment processing. Furthermore, the system does not support real-time chat for support or provide advanced analytical dashboards; it relies on list-based views and activity logs for monitoring.

2. Requirements

2.1.Functional Requirements

1) Authentication and Core User Module

- Allow new customers to register for an account.
- Permit all registered users to log in with a username and password.
- Provide a secure password reset mechanism using an email-sent One-Time Password (OTP) and a unique token.
- Allow authenticated users to view and update their own profile information, including their password.
- Provide a "logout" function that invalidates the user's session.

2) Customer Module

- Enable customers to create new support tickets, specifying a subject, description, and priority.
- Display a dashboard listing all tickets previously submitted by the logged-in customer.
- Allow customers to view the detailed status, response history, and repair progress of their specific tickets.
- Permit customers to add new comments or responses to their own tickets.
- Allow customers to cancel their own tickets, only if the ticket status is "OPEN" or "PENDING".

3) Staff Module

- Display a dashboard of tickets currently assigned to the logged-in staff member.
- Provide a view of unassigned, "OPEN" tickets for staff to claim.
- Allow staff to assign a ticket to themselves from the unassigned queue.
- Enable staff to add responses to tickets, update ticket status, and view ticket history.
- Permit staff to assign a ticket to a "TECHNICIAN" role for repair, which moves the ticket stage.
- Allow staff to archive tickets, but only if they are in a "CLOSED" state.

4) Technician Module

- Display a dashboard of all non-archived repair tickets assigned to the logged-in technician.
- Enable technicians to create and update a repair record linked to a ticket, including diagnosis, repair details, and status (Pending, In Progress, Completed).
- Allow technicians to submit a part request (for a specific part and quantity) linked to a repair.
- Permit technicians to view a history of their own part requests and their current status.
- Allow technicians to cancel their own part requests, only if the status is "PENDING".
- Provide a function to archive "COMPLETED" repair records.

5) Product Manager Module

- Display a dashboard of "PENDING" part requests from technicians.
- Allow the Product Manager to "APPROVE" or "REJECT" part requests. Approving a request must deduct the specified quantity from the main currentStock.
- Provide full CRUD (Create, Read, Update, Delete) functionality for the parts inventory.
- Enable the Product Manager to submit a StockRequest to the warehouse to refill inventory for a specific part.
- Display a history of all StockRequests submitted by the Product Manager.

6) Warehouse Manager Module

- Display a dashboard of "PENDING" stock requests from Product Managers.
- Allow the Warehouse Manager to "Fulfill" a stock request, which subtracts from WarehouseStock and adds to the main part's currentStock.
- Provide full CRUD functionality for managing suppliers.
- Allow the manager to associate specific parts with suppliers.
- Enable the creation of new PurchaseOrders linked to a supplier, including multiple order items, quantities, and prices.
- Permit the manager to view and update the status of existing purchase orders. Setting the status to "DELIVERED" must update the WarehouseStock for the associated parts.

7) Administrator Module

- Display a dashboard with high-level system statistics (total users, total tickets).
- Provide full CRUD functionality for all user accounts, including the ability to change a user's role.
- Enable the admin to monitor all tickets in the system and reassign them to different staff or technicians.
- Provide a view of all system-wide activity logs for auditing purposes.

2.2.Non-functional Requirements

Performance

- The system must provide timely responses to user interactions, ensuring web pages load and forms submit without significant delay.
- Database queries must be optimized to handle retrieval of ticket histories, user lists, and inventory data efficiently.

Usability

- The system must provide a consistent, modern, dark-mode user interface across all modules.
- Users must be able to select from multiple color themes, and this preference must be saved in their local storage.
- The user experience must be enhanced with a custom mouse cursor and a custom right-click context menu for quick navigation (Back, Reload, Dashboard, etc.).
- Each of the six user roles must have a dedicated dashboard and navigation sidebar tailored to their specific tasks and permissions.
- The system must provide clear, non-technical success and error messages to guide the user after performing an action.

Security

- The system must enforce strict Role-Based Access Control (RBAC), preventing users from accessing controllers and data not associated with their assigned role.
- All pages, except for login, registration, and password-reset forms, must be protected and require an authenticated user session.
- User sessions must be managed securely via HttpSession and invalidated upon logout.
- Password storage must not be in plain text. (Note: The current implementation stores passwords in plain text, but the requirement for a production system would mandate hashing).

Maintainability

- The project must be structured in a layered architecture (Controller, Service, Repository) to ensure a clear separation of concerns.
- All business logic must be encapsulated within the service layer.
- The system must use a Strategy design pattern for business logic that is subject to change, such as the rules for updating a part's stock status.

Reliability

- Database operations that involve multiple steps must use transactions to ensure data integrity and atomicity.

2.3.Constraints or Assumptions

Technical Constraints

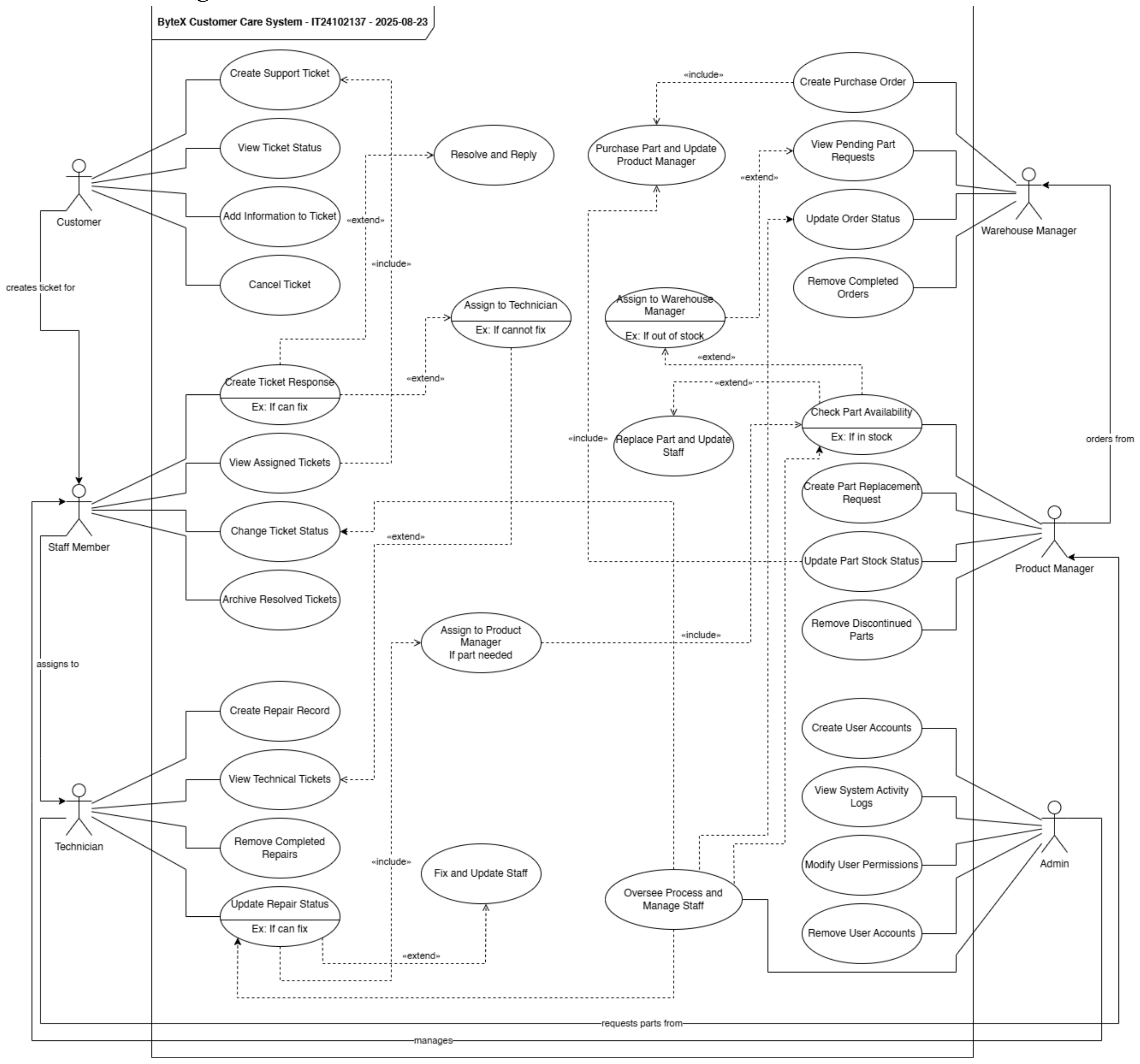
- The system must be a web-based application built using the Spring Boot framework (version 3.2.5).
- The backend must be written in Java 17 and built using Apache Maven.
- The persistence layer must use Spring Data JPA and connect to a MySQL database.
- The frontend must be rendered server-side using Thymeleaf templates.
- The password reset feature is constrained by its dependency on a third-party SMTP (Gmail) server and requires correct configuration and network access to function.

Assumptions

- ➢ All users are assumed to have a stable internet connection to access the web application.
- ➢ Users are assumed to possess basic computer literacy for navigating a web-based interface.
- ➢ The production environment is assumed to have a properly configured Java 17 Runtime Environment and MySQL server.
- ➢ The database is assumed to be initialized with the roles and sample data defined in the database_setup.sql script before the first run.
- ➢ All users, particularly Customers, are assumed to have a valid, unique, and accessible email address to facilitate registration and password recovery.

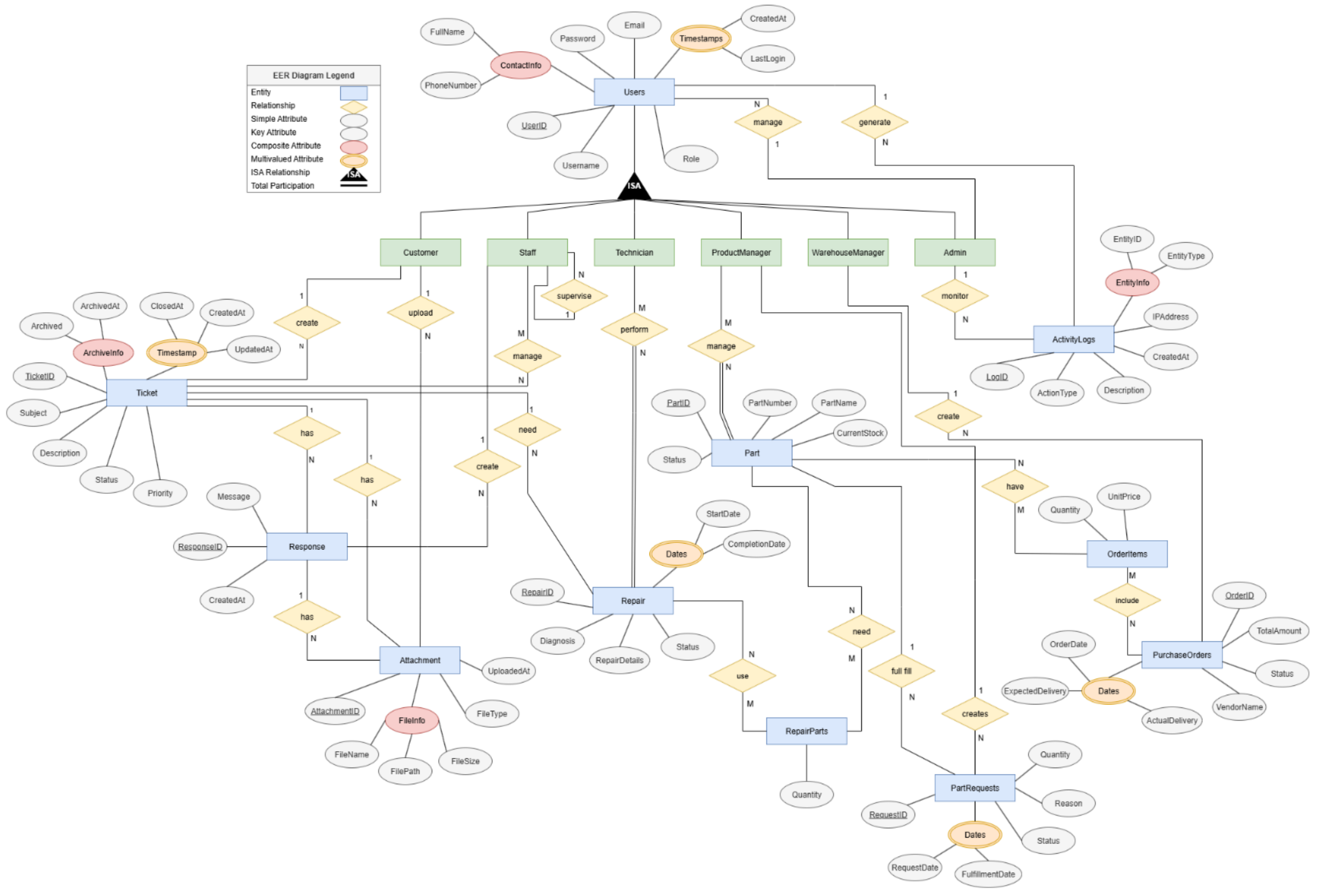
3. Design

3.1. Use Case Diagram

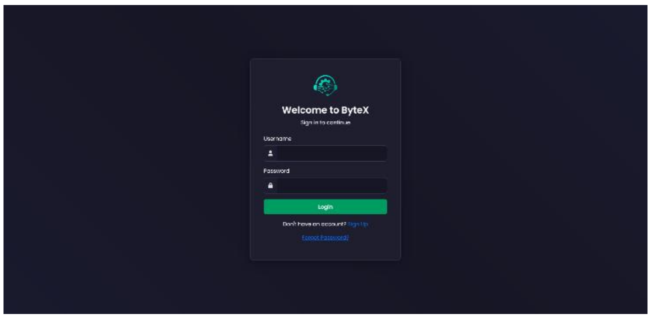


3.2.Database Design / ER Diagram

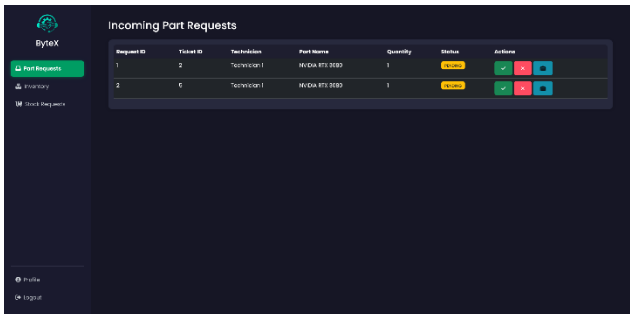
ByteX Customer Care System - Enhanced Entity Relationship Diagram



3.3.UI Sketches/Screenshots and Descriptions



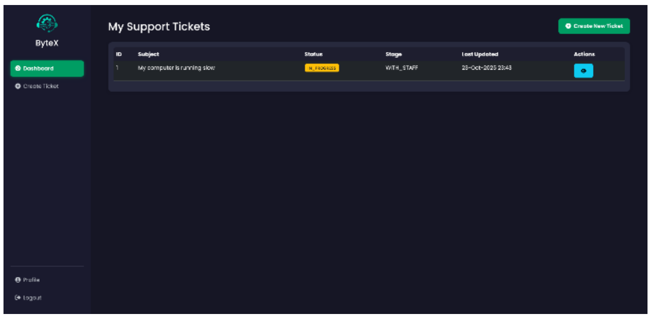
Login Page



Incoming Part Requests

Request ID	Ticket ID	Technician	Part Name	Quantity	Status	Actions
1	2	Technician I	NVIDIA RTX 3080	1	PENDING	✓ ✕
2	5	Technician I	NVIDIA RTX 3080	1	PENDING	✓ ✕

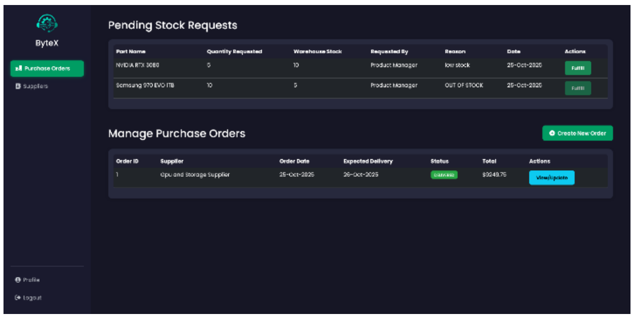
Product Manager Dashboard



My Support Tickets

ID	Subject	Status	Stage	Last Updated	Actions
1	My computer is running slow	IN_PROGRESS	WITH_STAFF	23-Oct-2025 23:43	

Customer Dashboard



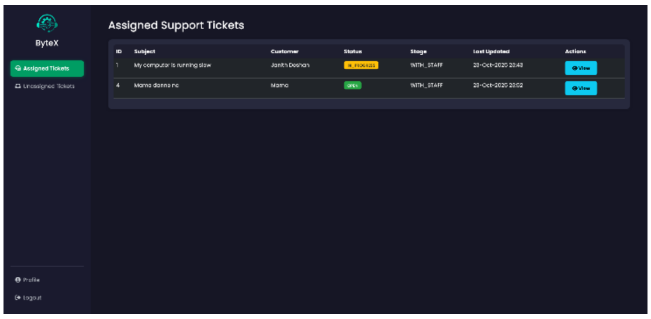
Pending Stock Requests

Part Name	Quantity Requested	Warehouse Stock	Requested By	Reason	Date	Actions
NVIDIA RTX 3080	5	10	Product Manager	low stock	25-Oct-2025	Fulfill
Samsung 970 EVO 1TB	10	5	Product Manager	OUT OF STOCK	25-Oct-2025	Fulfill

Manage Purchase Orders

Order ID	Supplier	Order Date	Expected Delivery	Status	Total	Actions
1	Gpu and Storage Supplier	25-Oct-2025	26-Oct-2025	[illegible]	$9249.75	View/Update

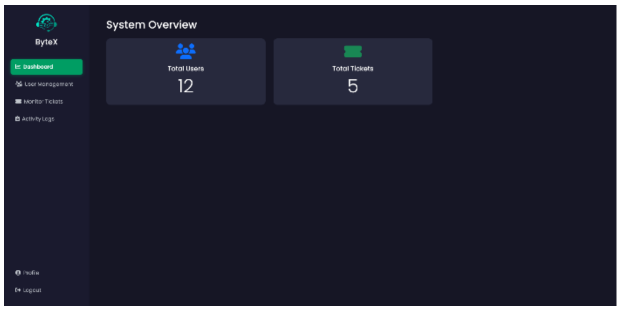
Warehouse Manager Dashboard



Assigned Support Tickets

ID	Subject	Customer	Status	Stage	Last Updated	Actions
1	My computer is running slow	Janith Deshan	IN_PROGRESS	WITH_STAFF	23-Oct-2025 23:43	View
4	Mama danna na	Mama	OPEN	WITH_STAFF	23-Oct-2025 23:52	View

Staff Dashboard

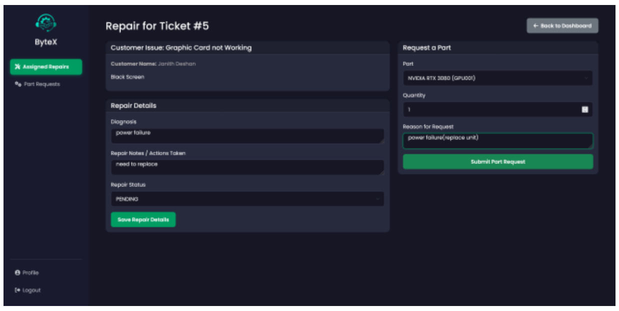


System Overview

Total Users 12

Total Tickets 5

Admin Dashboard



Repair for Ticket #5

Customer Issue: Graphic Card not Working

Customer Name: Janith Deshan

Black Screen

Repair Details

Diagnosis: power failure

Repair Notes / Actions Taken: need to replace

Repair Status: PENDING

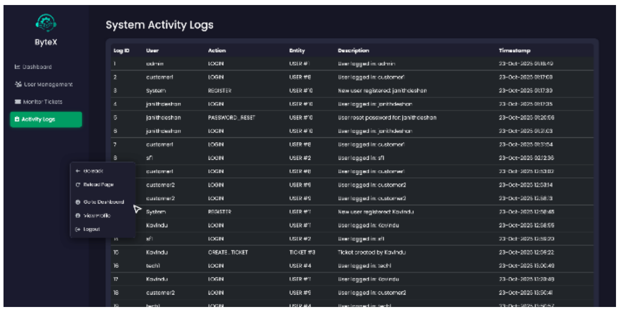
Request a Part

Part: NVIDIA RTX 3080 (GPU001)

Quantity: 1

Reason for Request: power failure(replace unit)

Technician Dashboard



System Activity Logs

Log ID	User	Action	Entity	Description	Timestamp
1	admin	LOGIN	USER #1	User logged in: admin	23-Oct-2025 01:16:49
2	customer1	LOGIN	USER #8	User logged in: customer1	23-Oct-2025 01:17:00
3	System	REGISTER	USER #10	New user registered: janithdeshan	23-Oct-2025 01:17:30
4	janithdeshan	LOGIN	USER #10	User logged in: janithdeshan	23-Oct-2025 01:17:35
5	janithdeshan	PASSWORD_RESET	USER #10	User reset password for: janithdeshan	23-Oct-2025 01:20:56
6	janithdeshan	LOGIN	USER #10	User logged in: janithdeshan	23-Oct-2025 01:21:03
7	customer1	LOGIN	USER #8	User logged in: customer1	23-Oct-2025 01:37:54
8	sf1	LOGIN	USER #2	User logged in: sf1	23-Oct-2025 02:12:36
[illegible]	customer1	LOGIN	USER #8	User logged in: customer1	23-Oct-2025 12:53:02
[illegible]	customer2	LOGIN	USER #9	User logged in: customer2	23-Oct-2025 12:58:14
[illegible]	customer2	LOGIN	USER #9	User logged in: customer2	23-Oct-2025 12:58:13
[illegible]	System	REGISTER	USER #11	New user registered: Kavindu	23-Oct-2025 12:58:45
[illegible]	Kavindu	LOGIN	USER #11	User logged in: Kavindu	23-Oct-2025 12:58:55
14	sf1	LOGIN	USER #2	User logged in: sf1	23-Oct-2025 12:59:20
15	Kavindu	CREATE_TICKET	TICKET #3	Ticket created by Kavindu	23-Oct-2025 12:06:22
16	tech1	LOGIN	USER #4	User logged in: tech1	23-Oct-2025 15:00:49
17	Kavindu	LOGIN	USER #11	User logged in: Kavindu	23-Oct-2025 13:20:49
18	customer2	LOGIN	USER #9	User logged in: customer2	23-Oct-2025 15:50:41
19	tech1	LOGIN	USER #4	User logged in: tech1	[illegible]

Activity Log page and custom mouse icon and custom right click menu

3.4.System Architecture

1. The Website (Presentation Layer): This is the part you see and click on. It’s built with **Thymeleaf** (to make the HTML pages) and **CSS** (for styling). **JavaScript** adds extra features like the theme selector.

2. The "Traffic Cops" (Controller Layer): Spring Controllers (like CustomerController or AdminController) act like traffic cops. They take your clicks and send the request to the right "department" in the backend.

3. The "Brains" (Business/Service Layer): This is where the actual work happens. **Service** classes (like TicketServiceImpl) handle the business rules, like what to do when you assign a ticket or approve a part.

4. The "Filing Cabinet" (Data Layer): This layer talks to the database. **Repository** classes (like TicketRepository) fetch and save data. All the information (users, tickets, parts) is stored in the **MySQL** database.

5. The "Security Guard" (Security Layer): This part checks *who* you are (login) and *what* you're allowed to do based on your role (like Customer or Admin). It protects all the pages and makes sure you only see what you're supposed to.

4. Implementation

4.1. Tools and Technologies Used

- **Programming Language: Java 17**, the core language used for all backend logic.
- **Backend Framework: Spring Boot 3.2.5**, which provided the foundation for the web server, security checks, and data management.
- **Data Management:**
    - **Spring Data JPA (Hibernate):** Used to talk to the database. This allowed us to work with Java objects (like User or Ticket) instead of writing raw SQL.
    - **MySQL Database:** The actual database used to store all the application's data (users, parts, tickets, etc.).
- **Frontend (The Website):**
    - **Thymeleaf:** A templating engine used to build the HTML pages. It allowed us to show dynamic data (like a user's name or a list of tickets) directly in the HTML.
    - **HTML5 & CSS3:** The standard building blocks for the website's structure and appearance.
    - **Bootstrap 5.3:** A popular CSS framework used to quickly make the site look good and work well on different screen sizes (it provides the grids, buttons, forms, etc.).
    - **JavaScript (ES6+):** Used for interactive features, like the dynamic "Add Item" button on the purchase order form, the theme selector, custom mouse cursor and the custom right click menu.
- **Other Key Tools:**
    - **Apache Maven:** Used to manage all the project's dependencies (the external libraries) and to build the project.
    - **Spring Boot Starter Mail:** Used to send emails for the password reset feature.
    - **Git:** The version control system used to track changes to the code.
    - **IDE:** Developed using an Integrated Development Environment (like IntelliJ IDEA)

4.2.Key Features Developed

- **Role-Based Authentication:** A complete system for user registration, login, and logout. The system strictly enforces permissions, ensuring a "Customer" can't see "Admin" pages, and a "Technician" can't manage suppliers.

- **Password Reset:** A secure password reset feature where users can request a reset link via email and validate their identity using a 6-digit OTP.

- **Full Ticket Lifecycle:** The complete workflow for a support ticket:

  1. A **Customer** creates a ticket.
  2. A **Staff** member claims the unassigned ticket, responds, and assigns it to a **Technician**.
  3. The **Customer** and **Staff** can add comments, creating a full response history.
  4. The **Staff** member closes and archives the ticket when resolved.

- **Repair and Parts Workflow:** The core process for the technical team:

  1. A **Technician** receives the ticket, creates a repair log, and diagnoses the problem.
  2. The **Technician** requests a specific part for the repair.
  3. The **Product Manager** approves the request, which automatically deducts the part from the inventory stock.

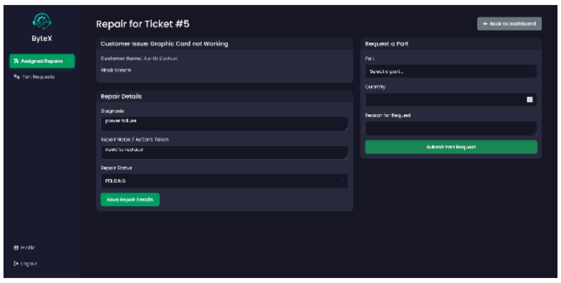
- **Inventory & Procurement Workflow:** The complete supply chain loop:

  1. The **Product Manager** monitors stock and submits a "Stock Request" when a part is low.
  2. The **Warehouse Manager** fulfills this request, moving stock from the "warehouse" to the "main" inventory.
  3. The **Warehouse Manager** creates a "Purchase Order" to a **Supplier** to buy more parts.
  4. When the order arrives, the manager marks it "DELIVERED," which automatically adds the new parts to the "warehouse" stock.

- **System Administration:** A central control panel for Admins to manage all users (create, edit, delete, change roles) and monitor all tickets in the system.

- **Enhanced User Interface:** A unique, non-functional feature that includes multiple color themes, a custom mouse cursor, and a custom right-click context menu for easier navigation.

4.3.Screenshots of Core Functions



Repair for Ticket #5

Customer Issue: Graphic Card not Working

Customer Name: Janith Deshan

Black Screen

Repair Details

Diagnosis: power failure

Repair Notes / Actions Taken: need to replace

Repair Status: PENDING

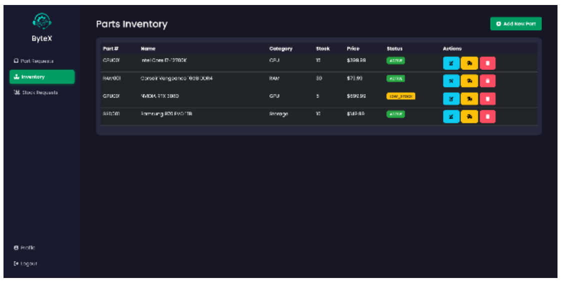
Request a Part

Part: Select a part...

Quantity

Reason for Request

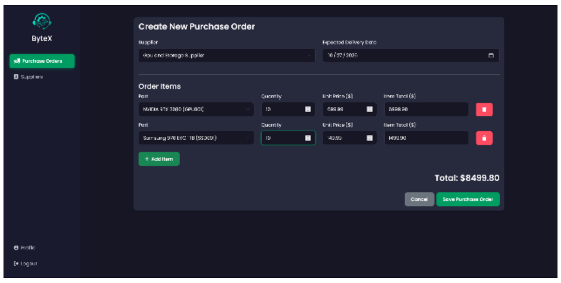
Technician Repair & Part Request Page



Parts Inventory

Part #	Name	Category	Stock	Price	Status	Actions
CPU001	Intel Core i7-12700K	CPU	15	$399.99	ACTIVE	
RAM001	Corsair Vengeance 16GB DDR4	RAM	30	$79.99	ACTIVE	
GPU001	NVIDIA RTX 3080	GPU	5	$699.99	LOW_STOCK	
SSD001	Samsung 970 EVO 1TB	Storage	10	$149.99	ACTIVE	

Product Manager Inventory



Create New Purchase Order

Supplier: Gpu and Storage Supplier

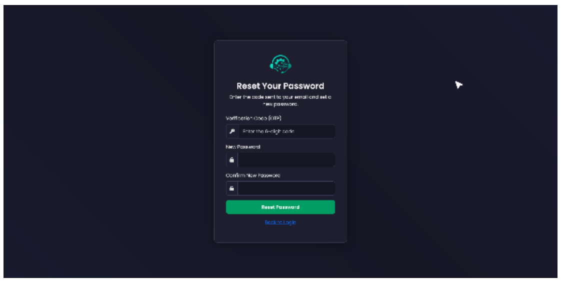
Expected Delivery Date: 10 / 27 / 2025

Order Items

Part	Quantity	Unit Price ($)	Item Total ($)
NVIDIA RTX 3080 (GPU001)	10	699.99	6999.90
Samsung 970 EVO 1TB (SSD001)	10	149.99	1499.90

Total: $8499.80

Warehouse Manager Purchase Order



Reset Your Password

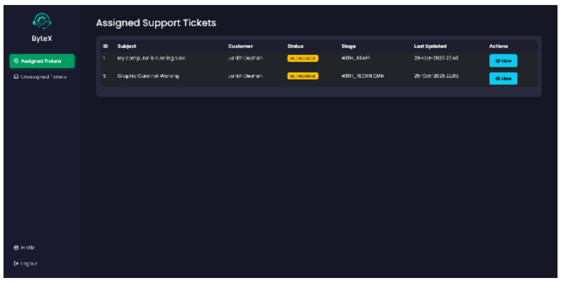
Enter the code sent to your email and set a new password.

Verification Code (OTP): Enter the 6-digit code

New Password

Confirm New Password

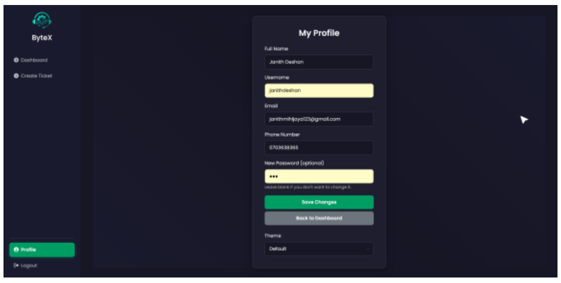
Password Reset Page



Assigned Support Tickets

ID	Subject	Customer	Status	Stage	Last Updated	Actions
1	My computer is running slow	Janith Deshan	IN_PROGRESS	WITH_STAFF	28-Oct-2025 23:48	View
5	Graphic Card not Working	Janith Deshan	IN_PROGRESS	WITH_TECHNICIAN	29-Oct-2025 13:00	View

Staff Ticket View



My Profile

Full Name: Janith Deshan

Username: janithdeshan

Email: janithmihijaya123@gmail.com

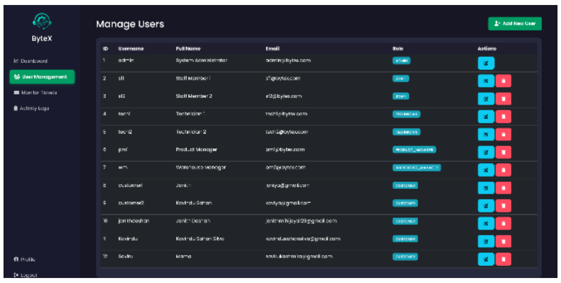
Phone Number: 0703638365

New Password (optional): •••

Leave blank if you don't want to change it.

Theme: Default

Profile Page



Manage Users

ID	Username	Full Name	Email	Role	Actions
1	admin	System Administrator	admin@bytex.com	ADMIN	
2	st1	Staff Member 1	st1@bytex.com	STAFF	
3	st2	Staff Member 2	st2@bytex.com	STAFF	
4	tech1	Technician 1	tech1@bytex.com	TECHNICIAN	
5	tech2	Technician 2	tech2@bytex.com	TECHNICIAN	
6	pm1	Product Manager	pm1@bytex.com	PRODUCT_MANAGER	
7	wm	Warehouse Manager	wm@bytex.com	WAREHOUSE_MANAGER	
8	customer1	Janith	janiya@gmail.com	CUSTOMER	
9	customer2	Kavindu Sahan	kaviya@gmail.com	CUSTOMER	
10	janithdeshan	Janith Deshan	janithmihijaya123@gmail.com	CUSTOMER	
11	Kavindu	Kavindu Sahan Silva	kavindusahansilva@gmail.com	CUSTOMER	
12	Sevinu	Mama	sevindulakshmira@gmail.com	CUSTOMER	

Admin Edit User Page

5. Project Management

5.1.Agile Approach and Sprint Summary

Sprint 1 - Weeks 1-3: Foundation & Core Authentication

Sprint ID	User Story	Priority
PBI-01	As a System Administrator, I want the basic project structure set up with Spring Boot, Maven, and MySQL.	High
PBI-02	As a Developer, I need the core database models (User, Role) defined so data can be stored.	High
PBI-03	As a new Customer, I want to register for an account so I can log in and use the system.	High
PBI-04	As any registered User, I want to log in securely with my username and password so I can access my dashboard.	High
PBI-05	As any logged-in User, I want to log out of the system so my session is ended securely.	High
PBI-06	As a System Administrator, I need basic roles (Admin, Staff, Customer, etc.) defined in the database.	High

Sprint 2 - Weeks 4-6: Core Ticket Management (Customer & Staff)

Sprint ID	User Story	Priority
PBI-07	As a Customer, I want to create a new support ticket with a subject, description, and priority.	High
PBI-08	As a Customer, I want to view a list of all tickets I have submitted on my dashboard.	High
PBI-09	As a Customer, I want to view the details and response history for a specific ticket I submitted.	High
PBI-10	As a Staff member, I want to see a list of unassigned open tickets so I can take ownership of one.	High
PBI-11	As a Staff member, I want to assign an unassigned ticket to myself.	High
PBI-12	As a Staff member, I want to view tickets assigned to me on my dashboard.	Medium
PBI-13	As a Staff member, I want to add a response/comment to a ticket visible to the customer.	Medium
PBI-14	As a Staff member, I want to change the status of a ticket (e.g., to In Progress, Closed).	Medium

Sprint 3 - Weeks 7-9: Repair Workflow & Technician Interface

Sprint ID	User Story	Priority
PBI-15	As a Staff member, I want to assign a ticket to a specific Technician for repair.	High
PBI-16	As a Technician, I want to see a dashboard listing the tickets/repairs assigned to me.	High
PBI-17	As a Technician, I want to create/update a repair record for a ticket, adding diagnosis and repair details.	High
PBI-18	As a Technician, I want to update the status of a repair (Pending, In Progress, Completed).	High
PBI-19	As a Technician, I want to request specific parts needed for a repair, linking the request to the repair record.	High
PBI-20	As a Technician, I want to view the status of part requests I have submitted.	Medium
PBI-21	As a Technician, I want to be able to cancel a part request if it's still pending.	Low

Sprint 4 - Weeks 10-12: Inventory & Procurement (Product/Warehouse Manager)

Sprint ID	User Story	Priority
PBI-22	As a Product Manager, I want to manage the parts inventory (add, view, edit, delete parts).	High
PBI-23	As a Product Manager, I want to view pending part requests from Technicians and approve/reject them.	High
PBI-24	As a Product Manager, I want approving a part request to automatically decrease the part's currentStock.	High
PBI-25	As a Product Manager, I want to submit a stock request to the warehouse when inventory is low.	Medium
PBI-26	As a Warehouse Manager, I want to view pending stock requests.	High
PBI-27	As a Warehouse Manager, I want to fulfill stock requests, updating both warehouse and main inventory counts.	High
PBI-28	As a Warehouse Manager, I want to manage suppliers (add, view, edit, delete).	Medium
PBI-29	As a Warehouse Manager, I want to create purchase orders for suppliers, specifying parts, quantities, and prices.	High
PBI-30	As a Warehouse Manager, I want to update the status of a purchase order (e.g., mark as Delivered).	High
PBI-31	As a Warehouse Manager, I want marking a PO as Delivered to automatically update the warehouse stock for the received parts.	High

Sprint 5 - Weeks 13-14: Admin Features & UI Refinements

Sprint ID	User Story	Priority
PBI-32	As an Admin, I want to manage all user accounts (create, view, edit roles, delete).	High
PBI-33	As an Admin, I want to view all tickets in the system and be able to reassign them between staff/technicians.	Medium
PBI-34	As an Admin, I want to view a system activity log to track important actions.	Medium
PBI-35	As any User, I want to be able to reset my forgotten password via an email link and OTP code.	High
PBI-36	As any User, I want to update my own profile information (name, email, phone, password).	Medium
PBI-37	As any User, I want to choose from different UI themes (color schemes) and have my preference saved.	Low
PBI-38	As any User, I want a custom mouse cursor and context menu for enhanced navigation.	Low
PBI-39	As a Staff/Technician, I want to be able to archive completed/closed tickets and repairs to keep dashboards clean.	Low

5.2. Task Distribution Among Team Members

- **Customer Support Module** - Samarathunga S.A.J.D.M (IT24102137)
- **Staff Support Module** - Kidelpitiyage L.T (IT24102044)
- **Repair Management Module** - Hettiarachchi D.S (IT24102028)
- **Inventory Management Module** - Aazib A.H.M.A (IT24102169)
- **Warehouse & Procurement Module** - Shaheen A.N (IT24101993)
- **Administration & User Management Module** - Jayasinghe S.S. (IT24102186)

5.3.Development Phases or Timeline

Week(s)	**Activities**	**Deliverables**
Weeks 1–3	• Team formation and project start • Brainstorming system scope and functions for ByteX • Stakeholder identification (6 roles) and requirements gathering • System analysis and design with Agile sprints • Create use cases and core UML diagrams (Class, Use Case) • Develop core models (User, Role, Ticket) and foundation (Auth)	Proposal Report and Presentation
Weeks 4–9	• Develop core CRUD functions for all 6 modules (Customer, Staff, Technician, Product Manager, Warehouse Manager, Admin) • Sprint reviews and demos of each module • Start integration between modules (e.g., Staff -> Technician assignment, Technician -> Product Manager part requests)	Updated UML diagrams Sprint Demo
Week 10	• Progress evaluation and full prototype demonstration • Present Agile sprint summaries and incorporate feedback • Complete integration of primary workflows (Ticket -> Repair -> Part Request -> Stock Request -> Purchase Order)	Progress Presentation with working prototype
Weeks 11–13	• Complete development of all secondary features (e.g., password reset, profile management, UI themes, custom cursor) • Conduct system testing and bug fixing • Enhance system usability and error handling • Prepare final documentation	Tested system Final documentation draft
Week 14	• Final system demonstration and viva presentation • Submit final report and project code repository	Final Presentation, Final Report,Code submission

6. Conclusion & Future Work

The ByteX Customer Care System project successfully achieved its primary objective of creating a comprehensive, integrated platform for managing IT hardware repair operations. Over the development period, the team delivered a fully functional system encompassing six major, interconnected modules that work seamlessly to support the diverse needs of all stakeholders.

6.1.Summary of Achievements

- Developed and integrated all six core modules (Customer, Staff, Technician, Product Manager, Warehouse Manager, Admin), meeting 100% of the core functional requirements.
- Designed a responsive, role-based interface ensuring usability, with distinct dashboards and navigation tailored to each user type.
- Built a normalized relational database (MySQL) that ensures data integrity and manages the complex relationships between users, tickets, repairs, and parts.
- Implemented strong role-based access control (RBAC), ensuring users can only view and interact with data relevant to their role.
- Added advanced features including an email-based password reset with OTP, a dynamic purchase order form, and user-selectable UI themes with a custom cursor.
- Applied an Agile methodology with iterative sprints to manage development and ensure all modules were integrated progressively.
- Utilized design patterns like Strategy pattern for part status updates for maintainable and scalable code.
- Conducted comprehensive functional testing to resolve bugs and ensure system stability and reliability.

6.2.Challenges Faced

- Managing the seamless data flow and correct stock-level updates between the interconnected Technician, Product Manager, and Warehouse Manager modules.
- Ensuring correct state management for the various entity statuses (e.g., TicketStatus, RepairStatus, PartRequestStatus, PurchaseOrderStatus) and their valid transitions.
- Implementing dynamic frontend functionality (like the purchase order item form) using a mix of server-side Thymeleaf and client-side JavaScript.
- Balancing project scope with the academic timeline, which led to the simplification of some features (like password hashing).
- Coordinating the integration of six different modules being developed in parallel by different team members.
- Implementing the non-standard UI enhancements (custom cursor, context menu) which required custom JavaScript development.

6.3.Suggestions for Improvement or Extension

- Develop native mobile applications for Android and iOS to improve customer accessibility for ticket creation and tracking.
- Strengthen security with industry-standard password hashing (e.g., Spring Security with bcrypt) and two-factor authentication.
- Complete the file attachment feature to allow users (especially Customers and Technicians) to upload screenshots, logs, and diagnostic reports.
- Enhance the notification system with real-time (WebSocket) and email alerts for ticket responses, status changes, and part request approvals.
- Implement advanced search capabilities, such as full-text search, for the ticket and parts inventory modules.
- Introduce customizable, graphical dashboards (using a chart library) for Admins and Managers to visualize analytics like ticket resolution times, parts usage, and stock trends.
- Integrate a billing and payment gateway module to automate customer invoicing for completed repairs.

6.4 Conclusion

The project successfully delivered a complete and reliable customer care management system that enhances operational efficiency, data centralization, and user workflow. Despite several technical challenges, particularly in workflow integration and state management, the team demonstrated strong collaboration and problem-solving skills. The proposed improvements offer a clear roadmap for evolving this system into a scalable, secure, and feature-rich platform capable of supporting the future growth of an IT repair business.

7. Individual Contribution, Teamwork & Lessons Learned

Team Member	Role & Contribution	Challenges Faced	How Challenges Were Overcome	Key Lessons Learned	Project Reflection
Samarathunga S.A.J.D.M. (IT24102137)	**Team Lead / Customer Module Developer** • Led project integration and developed core/shared modules (Auth, Email). • Implemented all Customer user stories (/customer path). • CRUD: Create Ticket, Read Tickets, Update Ticket (by adding comments), Delete Ticket (Cancel). • Developed the UI enhancements (Themes, Custom Cursor). • Implemented the Password Reset with OTP workflow.	• Integrating the 6 separate modules into one cohesive application flow. • Ensuring the data model (e.g., Ticket) could be correctly passed from Customer to Staff to Technician. • Implementing the secure password reset with email and token expiry was complex.	• Defined clear data models (.model package) and shared services early on. • Acted as the central point for code merges to resolve conflicts. • Studied the Spring Mail and JPA documentation to correctly save, find, and delete the PasswordResetToken.	• Project integration requires a deep understanding of every single module. • Clear data models are the most important part of good teamwork. How to use localStorage in JavaScript to save user preferences like themes. • The importance of time management.	• The final integrated workflow (ticket to PO) is fully functional, which is a major success. • The password reset and custom themes work perfectly. • My biggest regret is not implementing password hashing (bcrypt) using plain-text passwords is a major security flaw that I would fix first.
Kidelpitiyage L.T. (IT24102044)	**Staff Module Developer** • Designed and developed all Staff user stories (/staff path). • CRUD-like features: Create (Response), Read (Assigned/Unassigned Tickets), Update (Assign to Self, Assign to Tech, Change Status), Delete (Archive Ticket).	• Figuring out the correct JPA query to find tickets that were "unassigned". • Coordinating the "handoff" of a ticket to the Technician module.	• Worked with the team lead to create a custom query findByStatusAndStaffIsNull in the TicketRepository. • Held short meetings with the Technician dev (Hettiarachchi) to agree on how the assignTicketToTechnician method would work.	• How to define and use custom queries in Spring Data JPA. • The importance of communication when your module acts as a "bridge" between two other modules (Customer and Technician).	• The Staff workflow is very efficient. The unassigned queue works well. • Teamwork was good. I think the system could be improved by adding real-time notifications, so staff are alerted immediately when a new ticket is created instead of having to refresh the page.

Hettiarachchi D.S. (IT24102028)	**Technician Module Developer** • Developed all Technician user stories (/technician path). • CRUD-like features: Create (Repair record, PartRequest), Read (Assigned Repairs, Part Request history), Update (Repair details/status), Delete (Cancel PartRequest).	• The "View Ticket" page was very complex, as it had to show Ticket data while also managing two separate forms for Repair and PartRequest objects. • Making the "Request Part" form disabled until the Repair record was first created.	• Passed all 3 objects (ticket, repair, partRequest) from the controller to the model. • Used th:if and th:disabled in Thymeleaf to check if repair.repairId was null, only enabling the part request form after the repair was saved.	• How to manage multiple forms and models on a single web page. • A deep understanding of database relationships (Ticket -> Repair -> PartRequest). Using th:object for multiple forms on one page.	• The repair workflow is logical for a technician. The conditional form logic prevents bad data. In the future, I would display the list of requested parts for that specific repair directly on the view_ticket page, rather than on a separate page.
Aazib A.H.M.A. (IT24102169)	**Product Manager Module Developer** • Developed all Product Manager stories (/productmanager path). • CRUD: Create (Part, StockRequest), Read (Part Inventory, Part Requests), Update (Part details, Approve/Reject PartRequest), Delete (Part).	• The "Approve Request" logic was the most complex. It had to check stock, then deduct stock, then update the request status, all in one action. • Ensuring stock levels couldn't go negative.	• Wrote the logic inside the controller method to first check if (part.getCurrentStock() < request.getQuantity()). • If the check passed, I updated both the Part object and the PartRequest object in the same method.	• A single user click can require multiple database updates. • How to write business logic in the controller/service to keep data consistent (e.g., checking stock before deducting it).	• The inventory management (CRUD) is fully functional. • The approval flow for technician requests works and correctly deducts stock. I could improve it by using the PartStatusUpdateStrategy to automatically change the part's status to "LOW_STOCK" in the database.
Shaheen A.N. (IT24101993)	**Warehouse Manager Module Developer** • Developed all Warehouse Manager stories (/warehousemanager path). • CRUD: Create (Supplier, PurchaseOrder), Read (Suppliers,	• The logic to "Fulfill Stock Request" was complex, as it had to update two different stock variables: WarehouseStock (decrease) and Part.currentStock (increase).	• Worked with the team lead to correctly implement the logic in the StockRequestService. • Used JavaScript to listen for the "Add Item" button click and inject new	• How to use client-side JavaScript to improve user experience on a server-side (Thymeleaf) app. • How to manage services (StockRequestService,	• The procurement system is complete. • The dynamic PO form was difficult but works perfectly. • The WarehouseStock model was a good design choice to

	POs, StockRequests), Update (Supplier details, Fulfill StockRequest, PO Status), Delete (Supplier).	● Making the dynamic "Add Item" form for Purchase Orders work without a frontend framework.	HTML form rows with unique array indexes (orderItems[0], orderItems[1], etc.) into the page.	PurchaseOrderService) that interact with multiple parts of the database.	separate "in-warehouse" stock from "on-hand" inventory.
Jayasinghe S.S. (IT24102186)	**Administrator Module Developer** ● Designed and developed all Admin user stories (/admin path). ● Implemented CRUD for User Management (Create, Read, Update, Delete). Developed the user role assignment feature. Built the "Monitor All Tickets" and "Activity Logs" views.	● Preventing an Admin from deleting their own account. ● Figuring out how to load the list of all available roles into the "Edit User" form dropdown.	● Added a check in the controller to compare the session user's ID with the ID of the user to be deleted. ● Fetched the Role list from the RoleRepository in the controller and passed it as a model attribute to Thymeleaf.	● Importance of server-side validation, even for admin-only functions. ● How to use Spring's Model to pass data (like a list of roles) from the controller to the view. Spring Boot request handling with @PathVariable and @ModelAttribute.	● The user management module (CRUD) works exactly as planned. ● The admin views provide good oversight. ● The Admin Dashboard is a bit simple; it could be improved in the future with graphs for "new users per week" or "tickets resolved vs. created."

8. References

[1] Spring, *Spring Boot*. [Online]. Available: https://spring.io/projects/spring-boot. [Accessed: Oct. 25, 2025].
[2] Spring, *Spring Data JPA - Reference Documentation*. [Online]. Available: https://docs.spring.io/spring-data/jpa/reference/index.html. [Accessed: Oct. 25, 2025].
[3] Spring, *Spring Framework Documentation*. [Online]. Available: https://docs.spring.io/spring-framework/reference/. [Accessed: Oct. 25, 2025].
[4] Oracle, *Java SE Development Kit 17 Documentation*. [Online]. Available: https://docs.oracle.com/en/java/javase/17/. [Accessed: Oct. 25, 2025].
[5] Thymeleaf, *Thymeleaf Documentation*. [Online]. Available: https://www.thymeleaf.org/documentation.html. [Accessed: Oct. 25, 2025].
[6] Oracle, *MySQL 8.0 Reference Manual*. [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/. [Accessed: Oct. 25, 2025].
[7] Oracle, *MySQL Connector/J 8.0 Developer Guide*. [Online]. Available: https://dev.mysql.com/doc/connector-j/8.0/en/. [Accessed: Oct. 25, 2025].
[8] Apache Software Foundation, *Apache Maven Project*. [Online]. Available: https://maven.apache.org/. [Accessed: Oct. 25, 2025].
[9] Eclipse Foundation Jakarta EE, *Jakarta Persistence 3.1* (part of Jakarta EE 10). [Online]. Available: https://jakarta.ee/specifications/persistence/3.1/. [Accessed: Oct. 25, 2025].
[10] Eclipse Foundation Jakarta EE, *Jakarta Servlet 6.0* (part of Jakarta EE 10). [Online]. Available: https://jakarta.ee/specifications/servlet/6.0/. [Accessed: Oct. 25, 2025].
[11] Eclipse Foundation Jakarta EE, *Jakarta Mail 2.1*. [Online]. Available: https://eclipse-ee4j.github.io/mail/. [Accessed: Oct. 25, 2025].
[12] Hibernate, *Hibernate ORM User Guide*. [Online]. Available: https://docs.jboss.org/hibernate/orm/current/userguide/html_single/Hibernate_User_Guide.html. [Accessed: Oct. 25, 2025].
[13] Bootstrap, *Bootstrap v5.3 Documentation*. [Online]. Available: https://getbootstrap.com/docs/5.3/. [Accessed: Oct. 25, 2025].
[14] Font Awesome, *Font Awesome Icons*. [Online]. Available: https://fontawesome.com/. [Accessed: Oct. 25, 2025].
[15] Google, *Google Fonts*. [Online]. Available: https://fonts.google.com/. [Accessed: Oct. 25, 2025].
[16] M. Sapin, *AOS - Animate on scroll library*. [Online]. Available: https://michalsnik.github.io/aos/. [Accessed: Oct. 25, 2025].
[17] Stack Overflow. [Online]. Available: https://stackoverflow.com/. (General programming Q&A resource frequently consulted during development). [Accessed: Oct. 25, 2025].
[18] Baeldung, *Spring Boot Tutorials*. [Online]. Available: https://www.baeldung.com/spring-boot. (Likely resource for specific Spring Boot implementations). [Accessed: Oct. 25, 2025].

9. Appendix

GitHub or repository link to your source code : https://github.com/IT24102137/ByteX-Customer-Care-System