Sri Lanka Institute of Information Technology
Faculty of Computing

Information Technology Project - IT2080

80% Progress Report

Y2.S2.WD.IT.1.2

ITP24R_B1_12

Automotive Service Management System

	Name with Initials (Surname first)	Registration Number	Contact Phone Number	Email
1.	Perera S.A.B.M	IT22151292	0716823633	IT22151292@my.sliit.lk
2.	Wijekoon O.A	IT22265388	0713745565	IT22265388@my.sliit.lk
3.	Fonseka G.N.V.S	IT22207272	0710767697	IT22207272@my.sliit.lk
4.	Rusiru L.W.S	IT22220288	0717621250	IT22220288@my.sliit.lk
5.	Wickremathilaka J.A	IT22152114	0702031977	IT22152114@my.sliit.lk
6.	Kahawevithana S.D	IT22191342	0754503384	IT22191342@my.sliit.lk
7.	Nanayakkara G.L.C.S	IT22103918	0777220111	IT22103918@my.sliit.lk
8.	Abeywickrama W.D.S	IT22279798	0762882045	IT22279798@my.sliit.lk

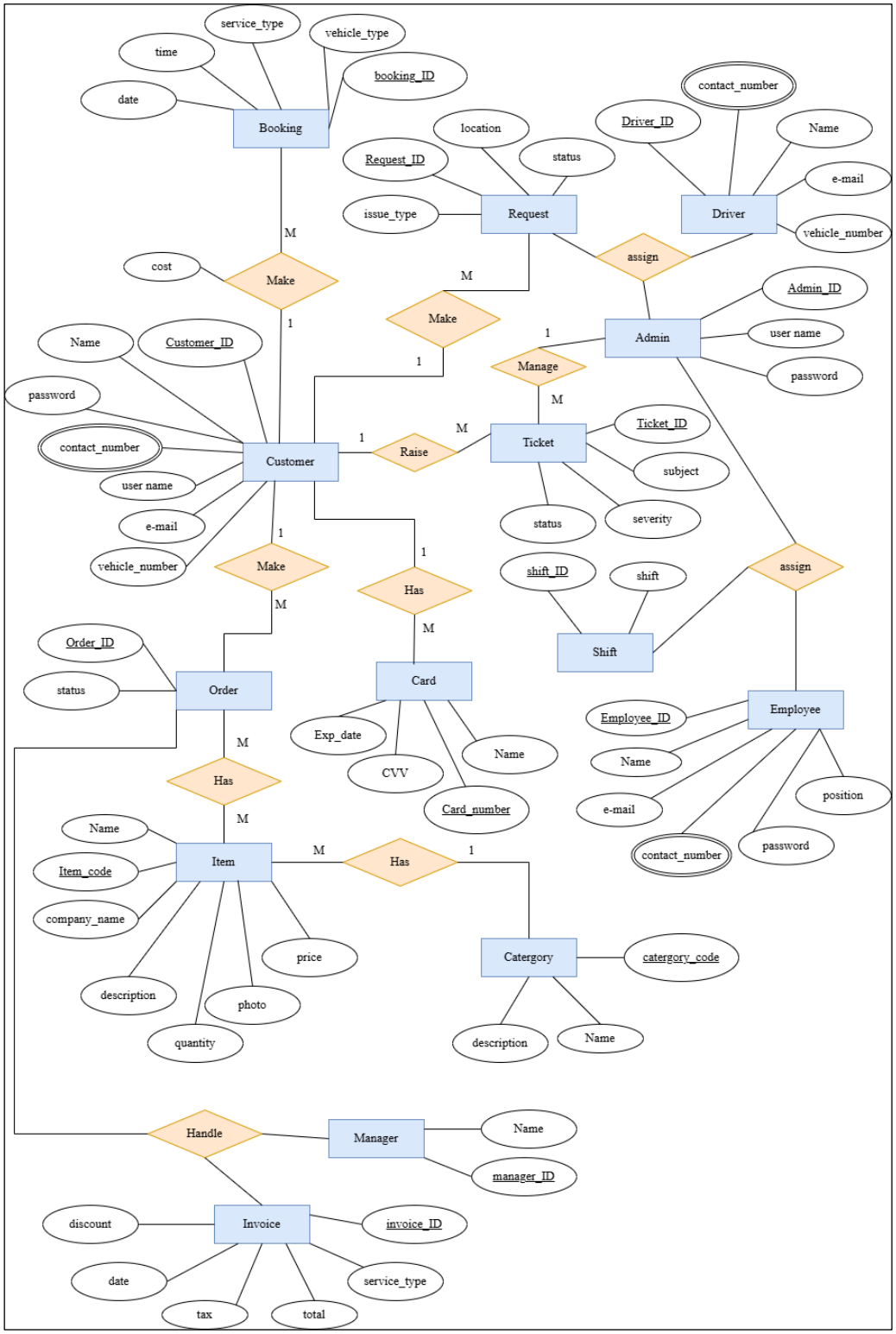
Submission date: 2024.09.23

Table of Contents

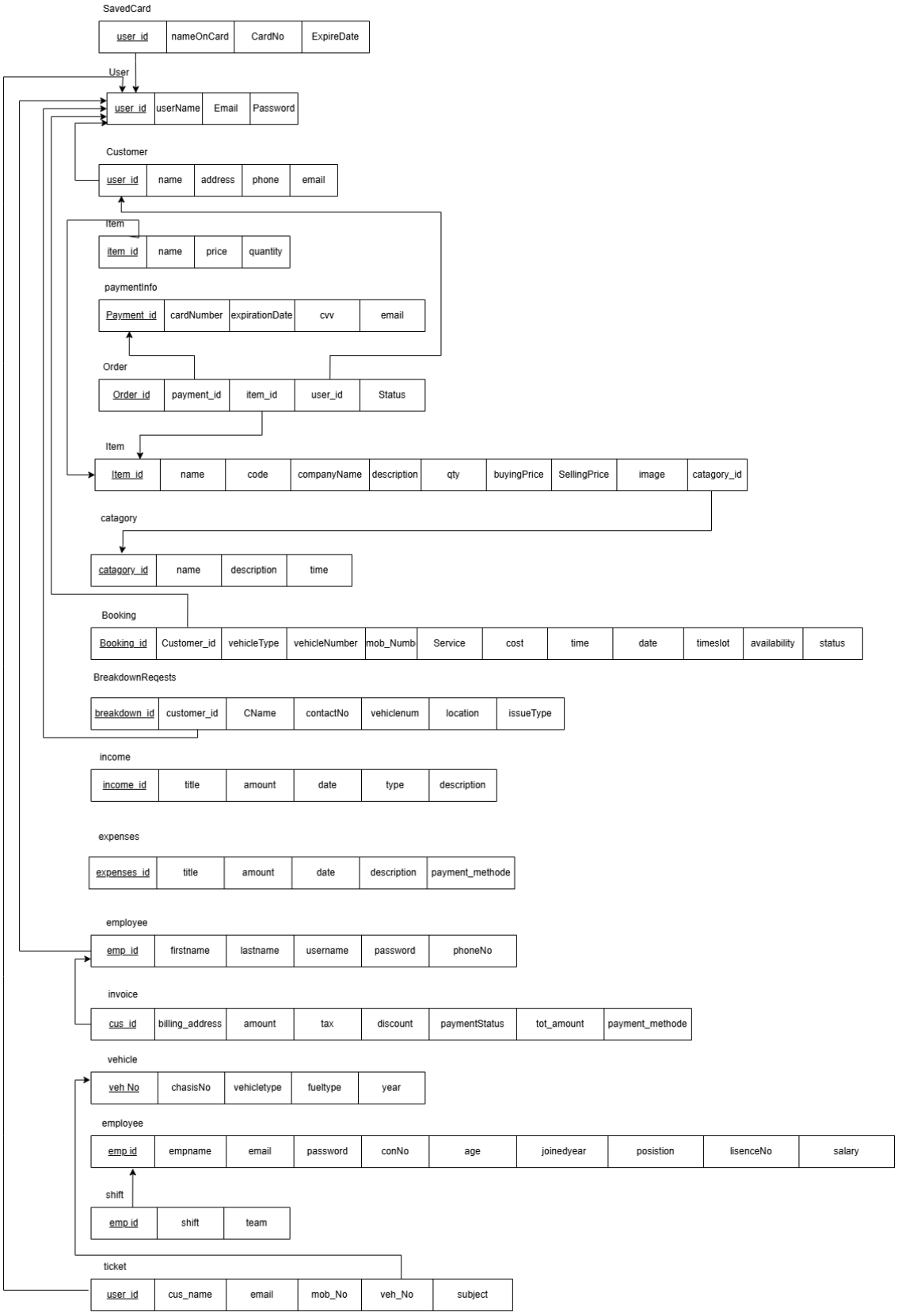
1. Respiratory Link

https://github.com/nisalfonseka/Vehicle-Service-System.git

2. ER Diagram



3. Normalization Schema



4. Network Design

4.1 Network Topology

A star topology will be employed by the system. This centralized method streamlines management and provides effective communication between components. All the devices will be connected via a single network switch:

- User Devices
- Backend Servers
- Database Server

Benefits of Star Topology:

- Simple to operate and diagnose.
- Scalable by expanding the switch's device count.
- High performance for connections between separate devices.

IP Addressing Scheme

A private IP address range will be used for internal network communication.

- Network Address: 10.0.0.0/24
- Usable IP Range: 10.0.0.1 -10.0.0.254
- Subnet Mask: 255.255.255.0

Backup plans and Redundancy

- Backend Servers: The backend program should be deployed across several load-balanced servers. If one server fails, traffic is distributed, and service continuity is guaranteed.

- Database Server: In this function that the primary server fails, preserve a backup copy of the data by implementing a database replication method.

4.2 Testing and Optimization

- Security Testing: To find weaknesses in the network and application security, use penetration testing. By being proactive, possible security threats are reduced.

Network Architecture

Automotive Service Management System utilizes a MERN stack architecture:

- Frontend (React): Runs on user devices and interacts with the backend via APIs.
- Backend (Node.js with Express): Handles application logic, database interaction, and API responses.
- Database (MongoDB): Stores all system data .

- (Optional) API Gateway: Acts as a single-entry point for frontend requests, routing them to appropriate backend services.

4.3 Data Flow

Data flows securely between components:

- The front end sends the backend HTTPS-preferable API calls.
- To facilitate efficient communication with the MongoDB database, the backend makes use of a connection pool.
- (optional) If used, the API Gateway routes queries.

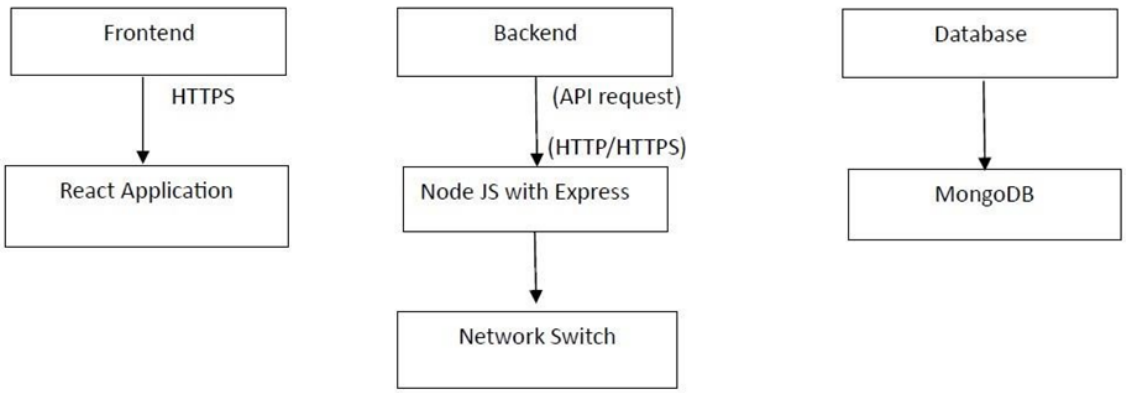
Security Measures

- HTTPS: Enforce HTTPS for all communication to encrypt data traffic.
- Authentication and Authorization: Implement mechanisms to authenticate users and grant access based on their roles.
- Database Security: Secure the MongoDB database with strong passwords, role-based access control, and regular backups.
- Firewalls: Implement firewalls to restrict unauthorized access to the network.

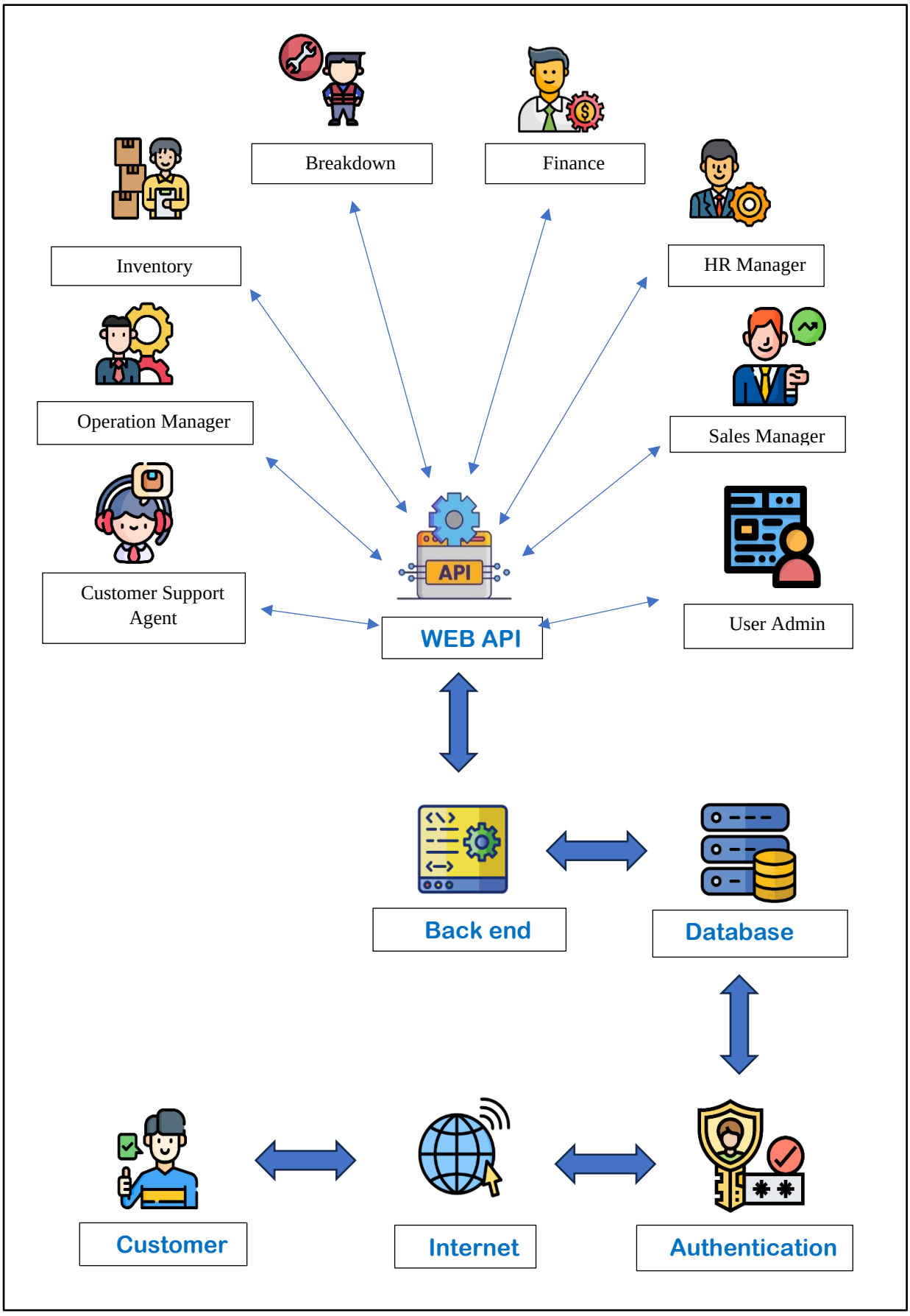
4.4 Scalability Considerations

- Load Balancing: Distribute incoming requests across multiple backend servers to handle increased user traffic.
- Cloud Deployment: Consider deploying the application on a cloud for easier scaling and resource management.

4.5 Network Diagram



5. High-level System Design Diagram



6. Internal Commercialization

This report outlines strategies for optimizing the value of the Automotive Service Management System for internal use. Although the system wasn't initially designed for external commercialization, it can enhance operational efficiency and potentially contribute to indirect revenue growth through improved service and streamlined internal processes.

Current Focus: Enhanced Operations and Efficiency

The system currently focuses on enhancing core operations and efficiency within the automotive service business. The following functionalities drive significant improvements:

- **Streamlined Booking Process**: Customers can easily book service appointments or request vehicle breakdown assistance.
- **Efficient Customer Support**: The system enables raising support tickets for faster resolution of customer queries or issues.
- **Employee Management System**: Tracks staff scheduling, attendance, and performance, improving human resource management.
- **Online Store & Inventory Management**: Simplifies the sale of parts and accessories, while maintaining accurate inventory levels.
- **Payroll Management:** Automates employee payroll, reducing administrative burden and errors.

Strategies for Maximizing Internal System Value

1. Comprehensive User Training:

- Ensure all employees, from service advisors to warehouse staff, are proficient in using the system.
- Conduct ongoing training to familiarize users with new functionalities or updates, ensuring optimal adoption and utilization.

2. System Customization:

- Tailor features to meet the unique workflow of your automotive service business, such as creating custom service categories or ticketing priorities.
- Customize dashboards to display KPIs like average service time, appointment frequency, and ticket resolution times for better management insight.

3. Performance Analytics and Reporting:

- Leverage data to monitor key performance indicators (KPIs) such as appointment rates, customer satisfaction, parts availability, and employee productivity.
- Generate automated reports to assess service delivery, inventory turnover, and payroll accuracy. This can help identify bottlenecks or areas for process optimization.

4. Integration with Existing Systems:

- Consider integrating the system with existing accounting software, CRM tools, or external vendor platforms to streamline operations and reduce manual data entry.
- Connect with customer communication channels (e.g., email, SMS) for automated appointment reminders and service updates.

Indirect Revenue Generation (Optional)

- **Increased Customer Retention**: The improved customer experience, such as faster appointment scheduling and effective customer support, can result in higher customer loyalty and repeat business.
- **Improved Employee Productivity**: Automating tasks like payroll and inventory management frees up employees to focus on customer service or sales, potentially increasing service capacity.
- **Inventory Optimization**: Real-time inventory tracking ensures essential parts are always in stock, reducing delays and enhancing service efficiency, leading to better customer satisfaction.
- **Data-Driven Marketing:**
  Using anonymized service data, the business can identify trends and customer preferences to target marketing efforts or promotions, potentially increasing sales of high-demand services or products.

In conclusion, by leveraging these strategies, the automotive service management system can significantly enhance internal operations, improve the customer experience, and create avenues for indirect revenue generation, ultimately driving overall business success.

7. Progress of Individual Functionalities

7.1 Breakdown Service Management – IT22151292

7.1.1 Completion level of the features

Test ID	Test Case	Step	Description	Expected Result	Actual Result	Completed (Yes/No)
1	Manage Breakdown Requests	1.1	Customer adds a new breakdown request	Request submitted successfully, validated against all fields and displayed in the system.	Request submitted successfully, validated against all fields and displayed in the system.	Yes
		1.2	View breakdown request	Display all the requests in the admin dashboard.	Display all the requests in the admin dashboard.	Yes
		1.3	Assign a driver to the request	Driver assigned is displayed in the system.	Driver assigned is displayed in the system.	Yes
		1.4	Update breakdown request	Status updated successfully and reflected in the system	Status updated successfully and reflected in the system	Yes
		1.5	Delete a breakdown request	Breakdown request deleted from the system successfully.	Breakdown request deleted from the system successfully.	Yes
2	Manage customer communication	2.1	Send WhatsApp message regarding estimated arrival time.	WhatsApp message sent to the customer with the estimated arrival time.	WhatsApp message sent to the customer with the estimated arrival time.	Yes
		2.2	Show breakdown request progress	Progress bar reflects the current status	Progress bar reflects the current status	No

				of the service request.	of the service request.	
3	Manage charges and distance calculation	3.1	Calculate service charge based on location	Calculate the estimated service charge based on predefined rates.	Calculate the estimated service charge based on predefined rates.	Yes
		3.2	Display total estimated charge	Total estimated charge displayed before request submission.	Total estimated charge displayed before request submission.	Yes
4	Generate reports	4.1	Admin generates a report of all breakdown requests for a specific period.	Breakdown report generated and download successfully.	Breakdown report generated and download successfully.	Yes
5	Search Functionality	5.1	Search breakdown request by request ID and customer name.	Breakdown request retrieved successfully using the request ID and customer name.	Breakdown request retrieved successfully using the request ID and customer name.	Yes

Completed Features:

- Create breakdown requests.
- Edit, view and delete breakdown requests.
- Calculate estimated service charge and display.
- Assign a driver to a breakdown location.
- Send estimated arrival time message to the customer via WhatsApp.
- Search request by request ID.
- Generate reports on handled breakdown requests.

Work to be completed:

- Implement frontend and backend to display progress of the request.
- Finalize implementation of all features.

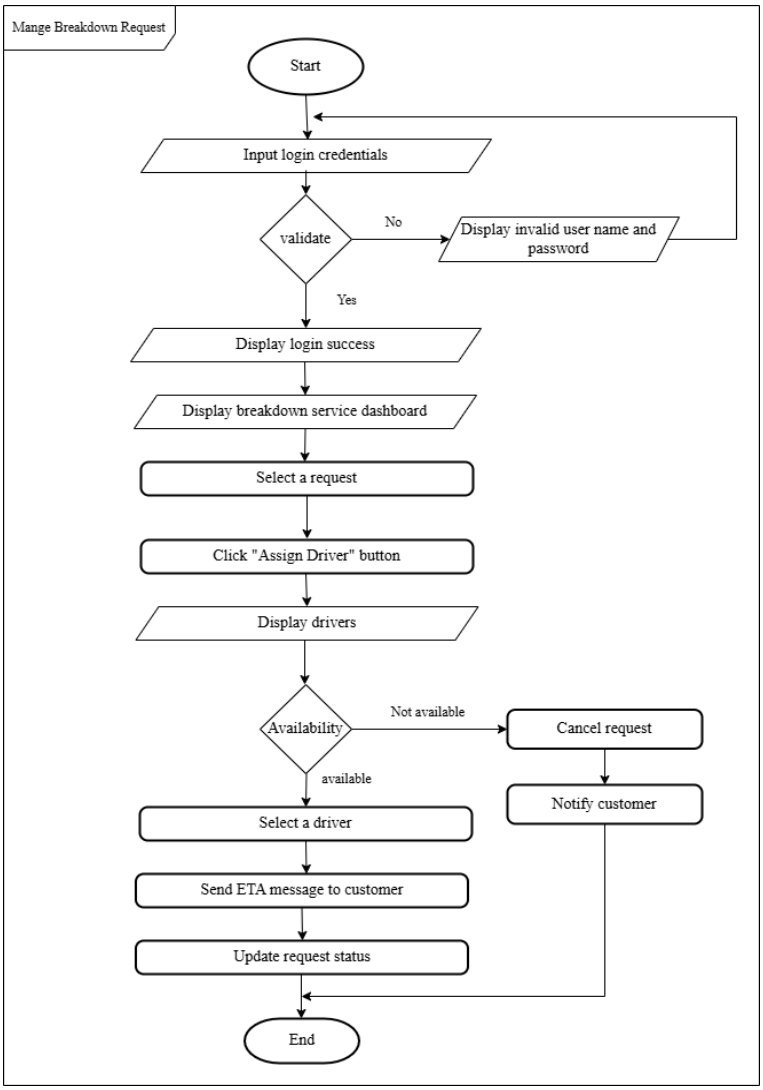
7.1.2 Innovative parts of the project

1. **ETA Message via WhatsApp**
   Send estimated time of arrival (ETA) to customers via WhatsApp once a driver is assigned. This provides real-time updates, increasing customer satisfaction.

2. **Map-Based Location Selection**
   Customers can select their exact location on an interactive map. The system automatically detects the location's GPS coordinates and geofences the service area to predefined zones for accurate distance calculation.

3. **Dynamic Service Charge Display Based on Real-Time Distance Calculation**
   The system calculates the distance between the customer's location and the service center in real time using OpenStreetMap. As the customer moves the pin to a different location, the total service charge dynamically updates, showing the base charge and additional fees for extra kilometers.

7.1.3 Flow Chart



7.1.4 Algorithms used

Assign Driver to Breakdown Request

```
const assignDriver = (request, availableDrivers) => {
   const assignedDriver = availableDrivers.find(driver => driver.isAvailable);
   if (assignedDriver) {
      assignedDriver.isAvailable = false; // Mark driver as unavailable
      return assignedDriver; // Return the assigned driver
   }
   return null; // No available driver
};
```

Calculate Service Charge Based on Distance

```
const calculateServiceCharge = (distance) => {
   const baseCharge = 25000; // Base charge for the first 10 km
   const additionalChargePerKm = 400; // Charge per additional km
   let totalCharge = baseCharge;

   if (distance > 10) {
      totalCharge += (distance - 10) * additionalChargePerKm;
   }

   return totalCharge.toFixed(2); // Return total charge rounded to 2 decimal places
};
```

7.1.5 Pseudo Code

Assign Driver to Breakdown Request

```
BEGIN
  FUNCTION assignDriver(request, availableDrivers)
     SET assignedDriver to find an available driver in availableDrivers
     IF assignedDriver exists THEN
        Mark assignedDriver as unavailable
        RETURN assignedDriver
     ELSE
        RETURN null // No available driver
     ENDIF
  END FUNCTION
END
```

Calculate Service Charge Based on Distance

```
BEGIN
  FUNCTION calculateServiceCharge(distance)
     SET baseCharge to 25000 // Base charge for the first 10 km
     SET additionalChargePerKm to 400 // Charge per additional km
     SET totalCharge to baseCharge

     IF distance > 10 THEN
        totalCharge += (distance - 10) * additionalChargePerKm
     ENDIF
     RETURN totalCharge rounded to 2 decimal places
  END FUNCTION
END
```

7.2 Customer Support Management – IT22265388

7.2.1 Completion level of the features

Test ID	Test Case	Step	Description	Expected Result	Actual Result	Completed (Yes/No)
1	Manage Customer Tickets	1.1	Customer raise a new support ticket	Ticket created successfully with all required details.	Ticket created successfully with all required details.	Yes
		1.2	Send email notification to customer upon ticket creation.	Customer receives an email confirmation of ticket creation.	Customer receives an email confirmation of ticket creation.	Yes
		1.3	View support tickets	View all support tickets on the admin dashboard.	View all support tickets on the admin dashboard.	Yes
		1.4	Update ticket status	Updated ticket status is reflected	Updated ticket status is	Yes

				on the dashboard.	reflected on the dashboard.	
		1.5	Delete a support ticket	Support ticket deleted successfully from the system.	Support ticket deleted successfully from the system.	Yes
2	Calculate resolution time	2.1	Calculate time taken for resolution based on issue type and severity.	Time taken for resolution calculated and displayed correctly.	Time taken for resolution calculated and displayed correctly.	Yes
3	Manage ticket resolution	3.1	Send resolution email to customer	Customer receives an email with the resolution of the ticket.	Customer receives an email with the resolution of the ticket.	Yes
4	Manage customer feedback	4.1	Customer rates the service after ticket resolution.	Customer submits a rating for the service received.	Customer submits a rating for the service received.	Yes
5	Generate reports	5.1	Generate support ticket report.	Downloadable report of all support tickets generated.	Downloadable report of all support tickets generated.	Yes
6	Search Functionality	6.1	Search tickets using ticket ID.	Successfully searched ticket by ID.	Successfully searched ticket by ID.	Yes

Completed Features:

- Raise support ticket.
- Edit, view and delete support ticket.
- Calculate estimated time for ticket resolution and display.
- Send email upon successful ticket creation.
- Search ticket by request ID.
- Generate reports on handled breakdown requests.

Work to be completed:

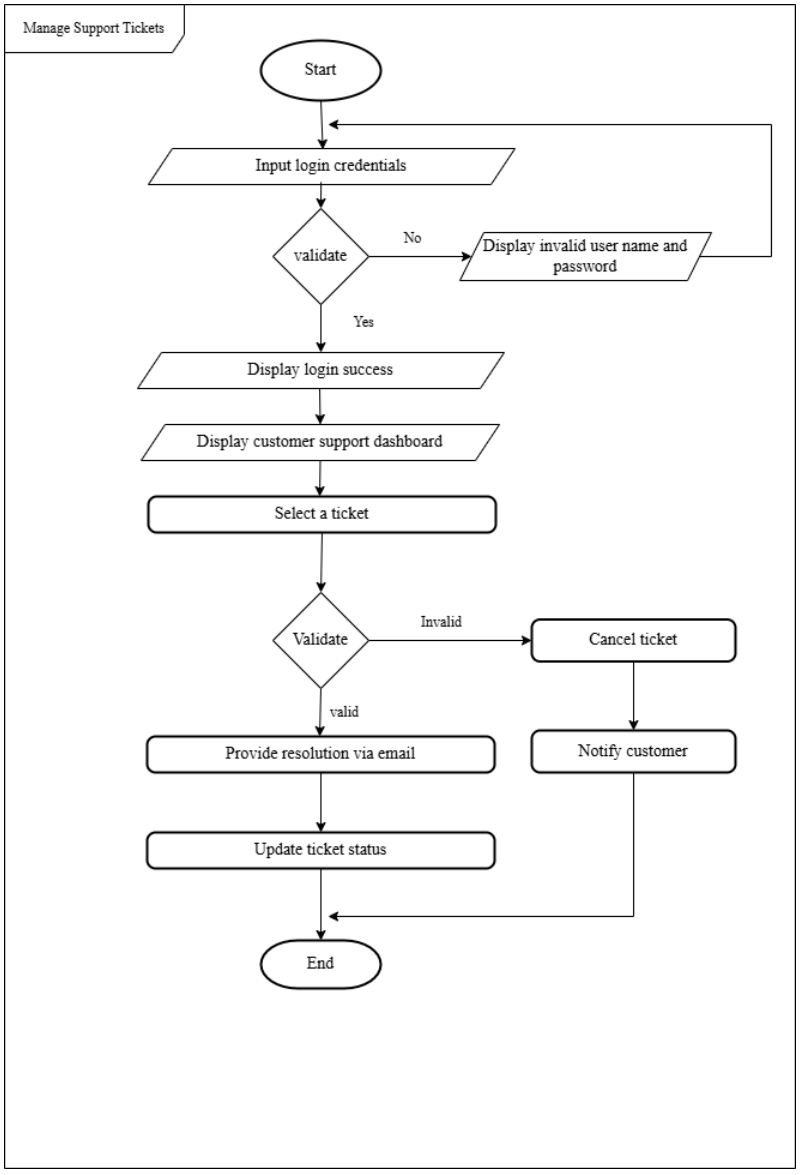
- Finalize implementation of all features.

7.2.2 Innovative parts of the project

1. **Intelligent Email Notifications**
   After a customer successfully creates a ticket, the system could send an automated, personalized email containing:
   Ticket Details
   Estimated Resolution Time: Based on the selected issue type and severity, the system can automatically calculate and include an estimated resolution time

2. **Time Estimation Based on Issue Type and Severity**
   For each issue type, the system could adjust the time estimates based on severity (high, medium, low priority) and display the time that will be taken to send a resolution.

7.2.3 Flow Chart



7.2.4 Algorithms used

Notification Algorithm to Send Emails

```
const sendEmailNotification = async (ticket) => {
   try {
      const emailResponse = await axios.post('/notifications/sendEmail', {
         to: ticket.customerEmail,
         subject: `Ticket ${ticket.ticketId} Update`,
         body: `Hello ${ticket.customerName}, your ticket status has been updated to ${ticket.status}.`
      });
      console.log("Email Sent:", emailResponse.data); // Log response
   } catch (error) {
      console.error("Error sending email notification:", error);
   }
};
```

Calculate Resolution Time Based on Severity

```
const calculateResolutionTime = (severity) => {
   switch (severity) {
      case 'High':
         return 2; // 2 hours
      case 'Medium':
         return 4; // 4 hours
      case 'Low':
         return 8; // 8 hours
      default:
         return 24; // Default to 24 hours for undefined severity
   }
};
```

7.2.5 Pseudo Code

Notification Algorithm to Send Emails

```
BEGIN
  FUNCTION sendEmailNotification(ticket)
     TRY
        SEND email to ticket.customerEmail
        SET subject to "Ticket {ticket.ticketId} Update"
        SET body to "Hello {ticket.customerName}, your ticket status has been updated to {ticket.status}."
        LOG "Email Sent"
     CATCH error
```

```
        LOG "Error sending email notification"
      ENDTRY
  END FUNCTION
END
```

Calculate Resolution Time Based on Severity

```
BEGIN
  FUNCTION calculateResolutionTime(severity)
    SWITCH severity
      CASE "High":
        RETURN 2 // 2 hours
      CASE "Medium":
        RETURN 4 // 4 hours
      CASE "Low":
        RETURN 8 // 8 hours
      DEFAULT:
        RETURN 24 // 24 hours for undefined severity
    ENDSWITCH
  END FUNCTION
END
```

7.3 Booking Management – IT22207272

7.3.1 Completion level of the features

Test ID	Test Case	Step	Description	Expected Result	Actual Result	Completed (Yes/No)
1	Manage Bookings	1.1	Customer adds a new booking by entering details.	Booking details entered successfully according to frontend validations.	Booking details entered successfully according to frontend validations.	Yes
		1.2	View booking details.	All booking details are displayed on the customer's booking page.	All booking details are displayed on the customer's booking page.	Yes

		1.3	Edit booking details (service type, date, time).	Modified booking details are displayed on the booking page.	Modified booking details are displayed on the booking page.	Yes
		1.4	Cancel an appointment.	The booking is successfully cancelled and removed from the system.	The booking is successfully cancelled and removed from the system.	Yes
2	Service Selection Calculation	2.1	Select a service (e.g., body wash, engine tune-up) and get cost.	The estimated cost and time for the selected service are calculated correctly.	The estimated cost and time for the selected service are calculated correctly.	Yes
3	Generate Booking Report	3.1	Generate a report on the bookings received in a day or month.	Download a PDF report with details of all bookings received in the selected time period (day or month), including service, date, and cost.	Download a PDF report with details of all bookings received in the selected time period (day or month), including service, date, and cost.	Yes
4	Search Functionality	4.1	Search for a customer by name in the booking system.	Successfully searched and retrieved customer booking details.	Successfully searched and retrieved customer booking details.	Yes
5	Search Functionality	5.1	Search for a vehicle by vehicle number or type.	Successfully searched and retrieved vehicle-	Successfully searched and retrieved vehicle-	Yes

				specific booking details.	specific booking details.	
6	WhatsApp Confirmation Message	6.1	Admin confirms or rejects the booking, and sends a message via WhatsApp.	The customer receives a WhatsApp message with booking confirmation details or rejection details including the reason for rejection.	The customer receives a WhatsApp message with booking confirmation details or rejection details including the reason for rejection.	Yes
7	View Past Bookings	7.1	Customer views past bookings in the profile page.	The customer's past bookings, including service details, are displayed in their profile page.	The customer's past bookings, including service details, are displayed in their profile page.	Yes

Completed Features:

1.Booking Management: 100% Complete

- Build frontend and backend for adding, editing, viewing, and deleting bookings.
- Create search option for finding bookings by customer name or date.
- Connect frontend and backend to store booking data in MongoDB.
- Set up automated SMS to confirm bookings with details.
- Design user-friendly interface for managing bookings.

2.Service Selection: 100% Complete

- Create a list for customers to choose services.
- Calculate and show estimated time and cost for services.
- Build easy-to-use frontend for selecting services.
- Connect backend to get service details from the database.
- Display selected services in the booking summary.

3.Automated Confirmation: 90% Complete

- Set up automated SMS for booking confirmations.
- Include service type, date, time, and location in messages.
- Create backend process to send SMS after booking.
- Show notifications to users about their confirmations.
- Ensure SMS delivery is reliable.

4.Appointment Modification: 80% Complete

- Allow users to edit appointment details and resubmit for approval.
- Implement cancellation option for customers to cancel appointments.
- Set up admin review for modified appointments.
- Design simple interface for editing and cancelling.
- Keep updates accurate in the booking system.

5.Report Generation:100% Complete

- Create reports for booked appointments with key information.
- Offer options to view, download, and print reports.
- Build backend logic to gather appointment data for reports.
- Make a user-friendly interface for accessing reports.
- Ensure reports are clear and easy to read.

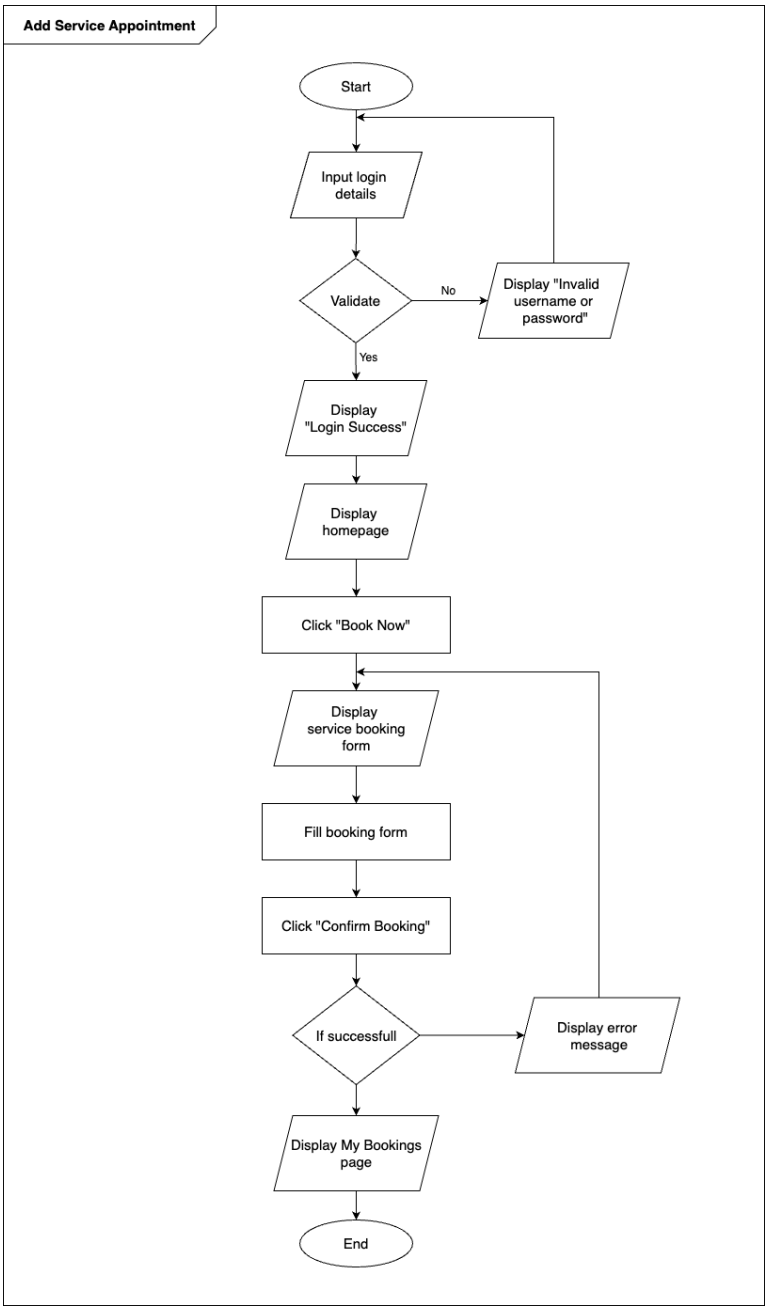
Work to be completed:

- Time Slot Management: Implement logic to allow only 3 bookings per time slot. Lock the time slot for further bookings once 3 users select it on the same day.

7.3.2 Innovative parts of the project

- When customers select the services they wish to have done, the booking form will automatically calculate and display the estimated cost and time for those selected services.
- Limiting bookings per time slot. when user select a time slot for the service, each time slot can only have 3 bookings, when 3 users select the same timeslot in a same day, that timeslot must be locked.

7.3.3 Flow Chart



7.3.4 Algorithms used

Search Algorithm

```
onChange={(e) =>
  setFilteredBooks(
    books.filter(
      (book) =>
        (book.customerName.toLowerCase().includes(e.target.value.toLowerCase()) ||
          book.vehicleNumber.toLowerCase().includes(e.target.value.toLowerCase())) &&
        (book.status === "New" || book.status === "Confirmed" || book.status === "Declined")
    )
```

```
  )
}
```

Data Fetching Algorithm

```
axios
  .get("http://localhost:5555/books")
  .then((response) => {
    const filteredBooks = response.data.data.filter(book => book.customerName === loggedInUser.username);
    setBooks(filteredBooks);
    setLoading(false);
  })
  .catch((error) => {
    console.log(error);
    setLoading(false);
  });
```

Service Selection and Cost Calculation

```
const handleCheckboxChange = (service) => {
  setSelectedServices((prevSelected) =>
    prevSelected.includes(service.name)
      ? prevSelected.filter((s) => s !== service.name)
      : [...prevSelected, service.name]
  );
  calculateTotalCostAndTime(service, !selectedServices.includes(service.name));
};

const calculateTotalCostAndTime = (service, isSelected) => {
  setTotalCost((prevCost) => prevCost + (isSelected ? service.cost : -service.cost));
  setTotalTime((prevTime) => prevTime + (isSelected ? service.time : -service.time));
};
```

7.3.5 Pseudo Code

Formatting Total Time

```
FUNCTION formatTime(minutes)
    hours ← FLOOR(minutes / 60)
    remainingMinutes ← minutes MOD 60
    RETURN CONCATENATE hours + "h " + remainingMinutes + "m"
END FUNCTION
```

Service Selection and Cost Calculation

```
FUNCTION handleCheckboxChange(service)
   IF service is already in selectedServices THEN
      REMOVE service from selectedServices
   ELSE
      ADD service to selectedServices
   END IF

   CALL calculateTotalCostAndTime(service, service was selected)
END FUNCTION

FUNCTION calculateTotalCostAndTime(service, isSelected)
   IF isSelected THEN
      INCREMENT totalCost by service.cost
      INCREMENT totalTime by service.time
   ELSE
      DECREMENT totalCost by service.cost
      DECREMENT totalTime by service.time
   END IF
END FUNCTION
```

7.4 Finance Management - IT22220288

7.4.1 Completion level of the features

Test ID	Test Case	Step	Description	Expected Result	Actual Result	Completed (yes/No)
1	Generate Invoices for the customers	1.1	Finance Manager Create the invoice	Entered details successfully according to frontend validations.	Entered details successfully according to frontend validations.	Yes
		1.2	View customer details. (Invoices)	Get all the details into Invoice List page.	Get all the details into Invoice List page.	Yes

		1.3	Edit invoices	Show all editeddetails in the Invoice List page	Show all edited details in the Invoice List page	Yes
		1.4	Delete Invoices	Successfully delete Invoices.	Successfully delete Invoices from the system.	Yes
2	Generate Invoice reports	2.1	Generate Invoice reports by clicking a button	Download pdf report	Download pdf report	yes
3	Search functionality in Invoice List page	3.1	Search invoice by Invoice name and Customer name.	Successfully searched Invoice.	Successfully searched Invoice.	yes
4	Generate Income by invoices List	4.1	Calculate the Total Amount including Taxes and Bonuses.	Successfully Calculate the Total Income.	Successfully Calculate the Total Income.	yes
5	Generate Income reports	5.1	Generate Income reports by clicking a button.	Download Income Statement pdf report.	Download Income Statement pdf report	yes
6	Search functionality in Income List page	6.1	Search Income by Customer name, Service Type and Date.	Successfully searched Invoices.	Successfully searched Invoices.	yes
7	Generate Expenses	7.1	Finance manager Can Add expenses	Entered details successfully according to frontend validations	Entered details successfully according to frontend validations.	yes

		7.2	View Expenses	Get all the details into Expenses List page.	Get all the details into Expenses List page.	yes
		7.3	Edit Expenses	Show all edited details in the Expenses List page.	Show all edited details in the Expenses List page.	yes
		7.4	Delete Expenses	Successfully delete Expenses.	Successfully delete Expenses.	yes
8	Search functionality in Expenses List page	8.1	Search Expenses by Title and Description.	Successfully searched Expenses.	Successfull ysearched Expenses.	yes
9	Generate Expenses by Expenses List.	9.1	Calculate the Total Amount of Expenses	Successfully Calculate the Total Expenses.	Successfully Calculate the Total Expenses.	yes
10	Generate Expense reports	10.1	Generate Expenses reports by clicking a button.	Download pdfreport.	Download pdf report.	yes

Completed Features:

- Create invoices to the customers.
- Edit, view and delete invoices.
- Search Invoices by Invoice name and Customer name.
- Generate PDF of Invoice List.
- Assign Invoices details to the Income table and view Income details.
- Search Incomes by Service Type and Date.
- Calculate the Total Amount Including taxes and Discounts in Income Table.
- Generate PDF of Income List.
- Create Expenses List.
- Edit, view and delete expenses.
- Search Expenses by Expense title and Description.

- Calculate Total Amount spent for Expenses
- Generate PDF of Expenses List.

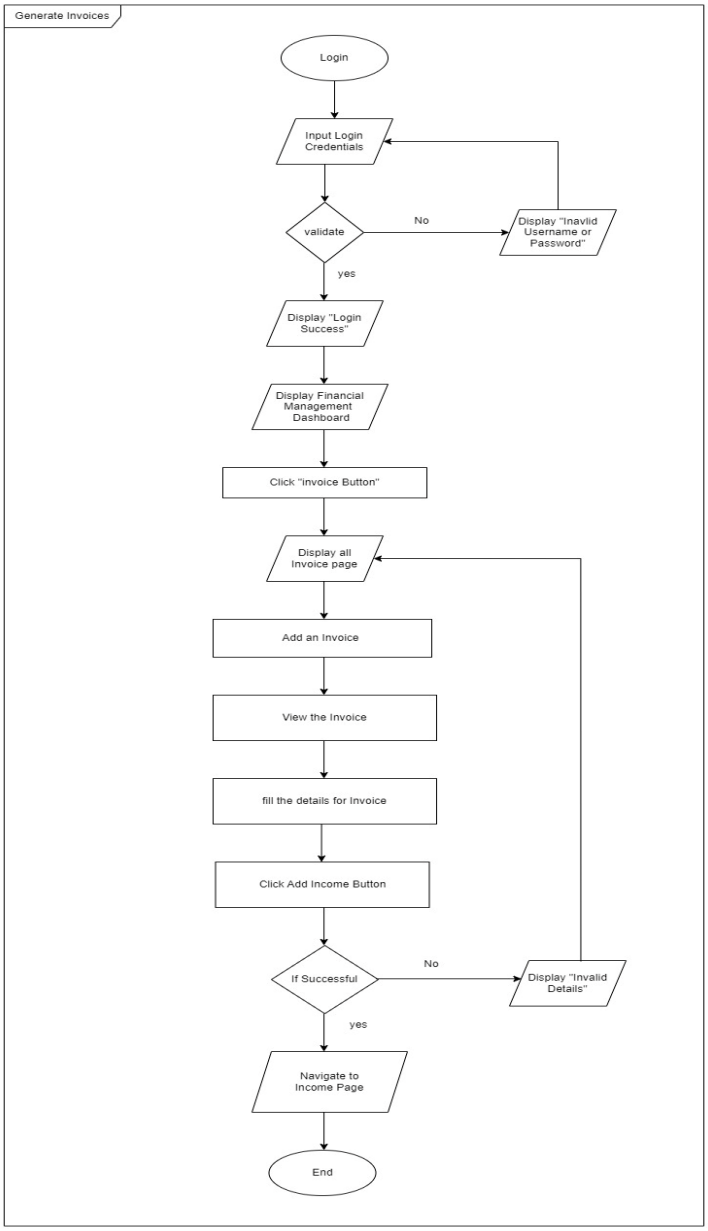
Work to be completed:

- Have to get Analysis Bar graph to the dashboard about Total Invoices, Total Incomes and Total Expenses.

7.4.2 Innovative parts of the project

- Adding a line chart to show the current total number of Invoices, Total number of Incomes and Expenses when updating them.

7.4.3 Flow Chart



7.4.4 Algorithms used

Search Invoices

```
const handleSearchChange = (event) => {
  setSearchTerm(event.target.value);
};

const filteredInvoices = invoices.filter((invoice) =>
  invoice.invoiceName.toLowerCase().includes(searchTerm.toLowerCase()) ||
  invoice.customerName.toLowerCase().includes(searchTerm.toLowerCase())
);
```

Filtering Invoices

```
const filteredInvoices = invoices.filter((invoice) =>
  invoice.invoiceName.toLowerCase().includes(searchTerm.toLowerCase()) ||
  invoice.customerName.toLowerCase().includes(searchTerm.toLowerCase())
);
```

Generate PDF for Invoices

```
 const generatePDF = (invoice) => {
  const doc = new jsPDF();

  doc.setFontSize(18);
  doc.text(`Invoice: ${invoice.invoiceName}`, 10, 10);
  doc.setFontSize(12);
  doc.text(`Customer: ${invoice.customerName}`, 10, 20);
  doc.text(`Service: ${invoice.serviceType}`, 10, 30);
  doc.text(`Date: ${new Date(invoice.invoiceDate).toLocaleDateString()}`, 10, 40);

  doc.autoTable({
    startY: 50,
    headStyles: { fillColor: [41, 128, 185], textColor: [255, 255, 255], fontSize: 12 },
    bodyStyles: { fontSize: 10 },
    alternateRowStyles: { fillColor: [240, 240, 240] },
    head: [['Description', 'Amount']],
    body: [
      ['Total Amount', `Rs. ${invoice.totalAmount.toFixed(2)}`],
      ['Tax Amount', `Rs. ${invoice.taxAmount.toFixed(2)}`],
      ['Discount Amount', `Rs. ${invoice.discountAmount.toFixed(2)}`],
      ['Final Amount', `Rs. ${invoice.calculatedTotalAmount.toFixed(2)}`],
      ['Payment Status', invoice.paymentStatus],
```

```
    ],
    margin: { top: 60 },
  });

  doc.text('Thank you! Come Again!', 80, doc.autoTable.previous.finalY + 10);
  doc.save(`${invoice.invoiceName}_Invoice.pdf`);
};
```

7.4.5 Pseudo Codes Used

Search function

```
function filterInvoices(invoices, searchTerm):
Initialize filteredInvoices as an empty array

For each invoice in invoices:
 Convert invoice.invoiceName to lowercase
 Convert invoice.customerName to lowercase
 Convert searchTerm to lowercase

  If searchTerm is in invoice.invoiceName or in invoice.customerName:
  Add the invoice to filteredInvoices

 Return filteredInvoices
```

Generate PDF for Invoices

```
function generatePDF(invoice):
Create a new PDF document

Set font size for title
Add text for "Invoice: " + invoice.invoiceName
Add text for "Customer: " + invoice.customerName
Add text for "Service: " + invoice.serviceType
Add text for "Date: " + invoice.invoiceDate (formatted)

Initialize table headers as ['Description', 'Amount']
Initialize table body with:
   ['Total Amount', invoice.totalAmount]
   ['Tax Amount', invoice.taxAmount]
   ['Discount Amount', invoice.discountAmount]
   ['Final Amount', invoice.calculatedTotalAmount]
   ['Payment Status', invoice.paymentStatus]
```

Generate a table using autoTable with headers, body, and custom styles

Add footer text: "Thank you! Come Again!"

Save the PDF file as invoice.invoiceName + "_Invoice.pdf"

7.3 Employee Management System – IT22152114

7.5.1 Completion level of the features

Test ID	Test Case	Step	Description	Expected Result	Actual Result	Completed (Yes/No)
1	Manage employee Records	1.1	Add new employees.	Entered details successfully according to frontend validations.	Entered details successfully according to frontend validations.	Yes
		1.2	View Employee Records.	Display all the employee records in Employee Manage page.	Display all the employee records in Employee Manage page.	yes
		1.3	Edit employee records.	Edit details In the edit page. And show them in the Employee Manage Page.	Edit details In the edit page. And show them in the Employee Manage Page.	yes
		1.4	Delete employee Records.	When press the delete button the record should be deleted.	Successfully delete employee from the system.	yes
2	Manage shift scheduler	2.1	Assign employees to Shifts scheduler.	Assign employees to ‘day’ or ‘night’ shift.	Assign employees to ‘day’ or ‘night’ shift.	yes

		2.2	View the Shift scheduler.	Should display the assigned employees in the shift scheduler.	Displays assigned employees.	yes
		2.3	Remove employees from Shifts scheduler.	When press delete button, the employee will be removed from the shifts scheduler.	Successfully delete employee from the Shift scheduler.	yes
3	Generate Employee report	3.1	Generate employee reports.	Download pdf report	Download pdf report	No
4	Search functionality	4.1	Search employee by employee name.	Relevant employee record will Display.	Relevant employee record will Display.	yes
5	Search functionality	5.1	Search employee by employee name or ID.	Searching Employees by their name or id to assign for Shift.	Searching Employees by their name or id to assign for Shift.	yes

Completed Features:

- Add new employees.
- View Employee Records.
- Edit employee records.
- Delete employee records.
- Assign employees to Shifts scheduler.
- View the Shift scheduler.
- Remove employees from Shifts scheduler.
- Search for employee by employee name.
- Search employee by employee name or ID.

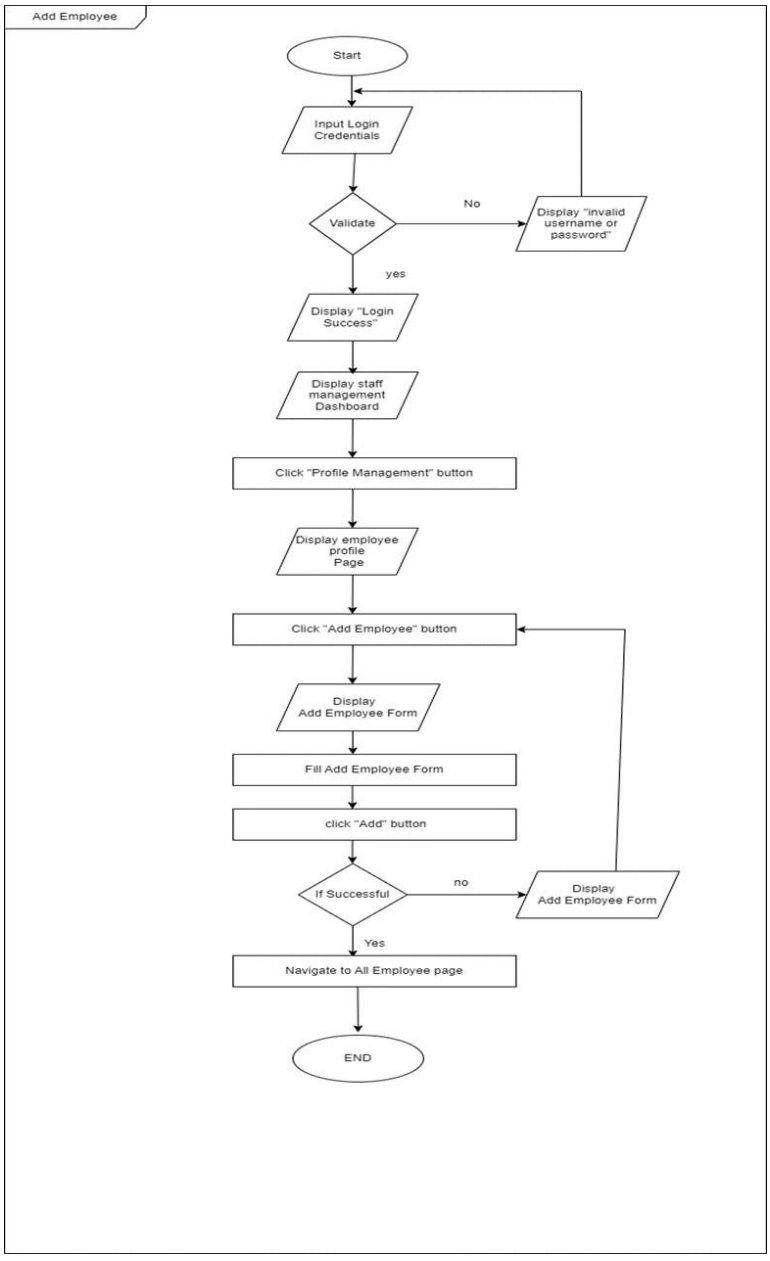
Work to be completed:

- Generate employee reports.

7.5.2 Innovative parts of the project

There is a feature in the dashboard that shows the total employee count and the total salary of the employees. When a new employee has been added it will automatically updates.

7.5.3 Flow Chart



7.5.4 Algorithms used

Fetching Employees for Day and Night Shifts

```
const fetchEmployees = async () => {
   try {
      const dayRes = await axios.get('/Shift/team?shift=Day');
      console.log("Day Shift Employees:", dayRes.data); // Log response
      const nightRes = await axios.get('/Shift/team?shift=Night');
      console.log("Night Shift Employees:", nightRes.data); // Log response
      setDayShiftEmployees(dayRes.data);
      setNightShiftEmployees(nightRes.data);
   } catch (error) {
      console.error("Error fetching shift employees:", error);
   }
};

useEffect(() => {
   fetchEmployees(); // Fetch employees when component is mounted
}, []);
```

Rendering Employee Rows for Each Shift

```
 const renderRows = (shiftEmployees) => {
   console.log("Rendering Rows for Shift Employees:", shiftEmployees);
   const rows = [];
   for (let i = 0; i < 5; i++) {
      if (i < shiftEmployees.length) {
         rows.push(
            <tr key={shiftEmployees[i].employeeId}>
               <td>{shiftEmployees[i].team}</td> {/* Display employee's team name */}
               <td>
                  <button className="delete-btn">Delete</button> {/* Placeholder for delete
*/}
               </td>
            </tr>
         );
      } else {
         rows.push(
            <tr key={i}>
               <td></td>
               <td></td>
            </tr>
         );
      }
```

```
    }
    return rows;
};
```

7.5.5 Pseudo Code

```
BEGIN
  FETCH employees for day and night shifts
  WHILE data is loading
      DISPLAY "Loading employees..."
  ENDWHILE

  IF data fetch is successful THEN
     DISPLAY "Day Shift Employees List"
     DISPLAY "Night Shift Employees List"
     Update state with fetched employees data
  ELSE
     DISPLAY "Error fetching shift employees"
  ENDIF
END
```

7.6 Inventory Management – IT22191342

7.6.1 Completion level of the features

Test ID	Test Case	Step	Description	Expected Result	Actual Result	Completed (Yes/No)
1	Add item	1.1	Open inventory management section	Inventory screen opens	Inventory screen opens as expected	yes
		1.2	Click on "Add Item" button	"AddItem" form opens	Form opens correctly	yes
		1.3	Fill in item details (name, category, stock, etc.)	Item details are entered	Fields accept input correctly	yes
		1.4	Save the item	Item is saved in the inventory	Item is saved and appears in inventory list	yes

2	Edit item	2.1	Open the inventory list	Inventory list is displayed	Inventory list appears correctly	yes
		2.2	Select an item to edit	Item details are shown	Item details are displayed for editing	yes
		2.3	Select an item to edit	Item details are shown	Modified details are accepted	yes
		2.4	Save the changes	Changes are saved and reflected in the inventory	Changes are saved successfully	yes
3	Remove item	3.1	Open the inventory list	Inventory list is displayed	Inventory list appears correctly	yes
		3.2	Select an item to remove	Item details are shown	Item details are displayed	yes
		3.3	Click on "Remove Item" button	Item is removed from the inventory	Item is removed successfully	yes
4	View Low Stock Items	4.1	Open inventory list and filter by low stock	Low stock items are displayed	Low stock items are shown correctly	yes
5	Search Item	5.1	Enter item name or ID in search bar	Relevant items are shown	Correct search results are displayed	yes
6	Update Stock Quantity	6.1	Open the inventory list and select an item	Item details are shown	Item details appear correctly	yes
		6.2	Modify the stock quantity	Updated quantity is entered	Updated quantity is accepted	yes

		6.3	Save the changes	Changes are saved and reflected in the inventory	Changes are saved successfully	yes

Completed Features:

1. **Stock Tracking:** Ability to track available parts in real-time, including part numbers, quantities, and locations.
2. **Reordering System:** Automatic reordering when stock levels fall below a predefined threshold.
3. **Inventory Reports:** Generate detailed reports on current stock, part usage trends, and stock valuation.
4. **User Access Control:** Role-based access to ensure that only authorized personnel can modify inventory.
5. **Alerts and Notifications:** Notify staff about low stock levels or critical shortages in certain parts.

Work to be completed:

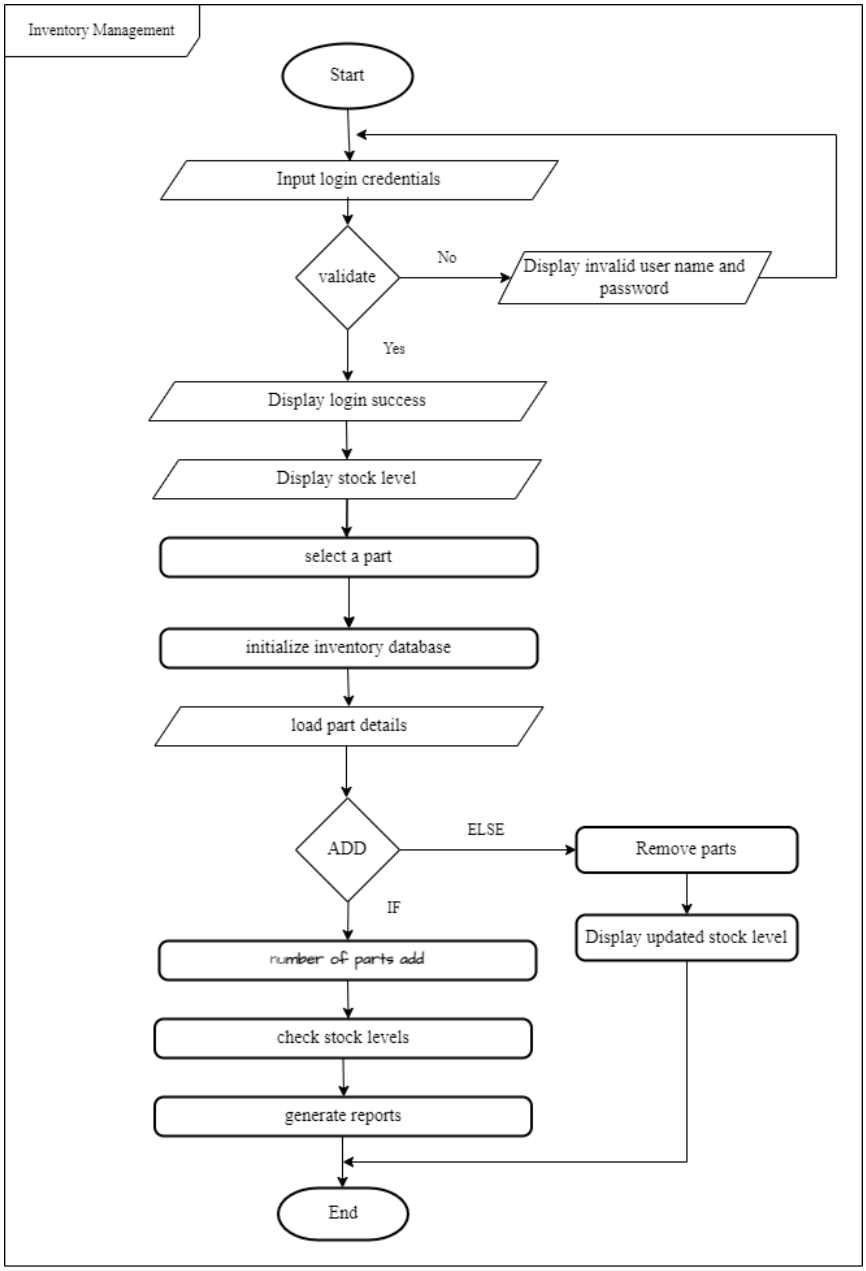
1. **Integration with Accounting System:** Connect the inventory system to an accounting platform for automatic invoicing and billing.
2. **Advanced Analytics:** Implement machine learning models to predict future stock requirements based on historical data.
3. **Warranty and Part Expiration Tracking:** Add the ability to track warranty status and expiration dates for specific parts.

7.6.2 Innovative parts of the project

1. Predictive Stock Reordering: Use machine learning algorithms to predict when parts will run low based on historical data and current trends.

2. Voice Commands for Inventory Updates: Allow users to update stock or check inventory through voice commands integrated with smart assistants.

7.6.3 Flow Chart



7.6.4 Algorithms used

Reordering Algorithm (Threshold-based)

- Constantly monitors stock levels and triggers an order to suppliers when a part falls below a predefined threshold.

7.6.5 Pseudo Code

Reordering Algorithm (Threshold-based)

```
For each part in inventory:
   If part.quantity < part.reorder_threshold:
      Trigger reorder process
      Notify supplier
```

Initialize inventory database

```
  - Load part details (Part ID, Name, Stock Level, Supplier Info, etc.)
```

Check inventory

```
  Input: Part ID or QR code scan
  IF Part ID exists in inventory THEN
    Display Part Details (Stock Level, Location, Supplier, etc.)
  ELSE
    Display "Part Not Found"
```

Add or remove parts

```
  Input: Add or Remove
  IF Add THEN
    Input: Number of parts to add
    Update stock level in database
    Display updated stock level
  ELSE IF Remove THEN
    Input: Number of parts to remove
    Update stock level in database
    Display updated stock level
```

Check stock levels

```
  FOR each part in inventory
    IF stock level < reorder threshold THEN
      Trigger reorder process
      Notify supplier
    ELSE
      Continue checking
```

Generate reports

```
  IF report requested THEN
    Select report type (Daily, Weekly, Monthly)
    Generate report from inventory data
    Display/Export report
```

7.7 Online Selling Management - IT22103918

7.7.1 Completion level of the features

Test ID	Test Case	Step	Description	Expected Result	Actual result	status
1	Browsing Items	1.1	Verify if the customer can log in and access the online store	Customer can log into system and go to store	Customer can login to system and go to store	Passed
2	Cart Handling	2.1	Select an in-stock item and click "Add to Cart."	Item add to cart	Item add to cart	passed
		2.2	Select out of-stock item	System display that item out of stock	Out of stock alert showing	passed
		2.3	Increase Item Quantity	Increase the quantity of an item in the cart	Increase the quantity of an item in the cart	passed
		2.4	Decrease Item Quantity	Decrease the item quantity in the cart	Decrease the item quantity in the cart	passed
		2.5	Remove Item in cart	Remove the item	Item removing	passed
3	Checkout	3.1	Proceed to Checkout	customer can proceed to the checkout page after selecting items.	Customer is directed to the checkout page with order summary	passed
			Calculate Total	Verify that the customer can view the total amount for the items they have selected.	System display the calculated total to customer	passed
		3.2	Cancel Order Click "Cancel."	Order canceled and navigate	Order canceled and navigate	passed

				back to the cart	back to the cart	
			Download Order Summary	Verify if the customer can download the order summary after placing the order	Summery report downloading	passed
4	Credit Card Management	4.1	Add a new credit or debit card to the system.	Verify if the customer can add a credit or debit card	Card details are saved	passed
		4.2	View added card	Verify customer can view the added cards	Customer can view cards	passed
			Add more than 3 cards.	Verify if the system restricts the customer to a maximum of 3 cards	System prevents adding more than 3 cards and the add button disable	passed
			Delete a saved credit or debit card.	Verify if the customer can delete a saved card	The selected card is removed from the system	passed
			. Update details of a saved card.	Verify if the customer can update the details of a saved card.	Card details are updated successfully, and the card is available for transactions.	passed
5	Inventory Management	5.1	Sales manager logs in and views inventory	Verify if the sales manager can view	Inventory displays item name, code, selling	passed

				the list of inventory items.	price, description, and available quantity.	
			Sales manager uses the search bar to find an item.	Verify if the sales manager can search for an item in the inventory.	Search results display the correct items based on the search query.	passed
6	Order Management	6.1	. Sales manager views orders in the system.	Verify if the sales manager can view all orders placed through the store	All orders, are shown.	passed
			. Sales manager filters orders by status Pending/Active/Cancelled	Verify if the sales manager can view pending, active, and canceled orders separately.	System correctly filters and displays orders based on their status.	passed
			Sales manager clicks "See Details" on an order.	Verify if the sales manager can view detailed order information	Sales manager can view order details, including items, total price, customer details, and status.	passed
			Accept Pending Order	Verify when click on the accept button order	Order status update to the pending status to accept status	passed

				status update according to the that		
			Cancel Order	Sales manager cancels an order	Order status updating to cancelled	passed
			Sales manager deletes an order from the system	Verify if the sales manager can delete unnecessary orders.	Selected order is successfully deleted from the system.	passed
			. Sales manager generates a sales report	Verify if the sales manager can generate a report on sales made in the online store.	System generates a sales report summarizing sales data within the selected date range or criteria.	passed

Work to be Completed:

1. **Implement Promotion Code System**:
   - When a customer uses a promo code, they should be able to receive a discount.

2. **Store Item Filter by Categories**:
   - Add functionality to filter store items based on categories.

3. **Divide Orders into Separate Pages by Status**:
   - Organize orders into different pages based on their status: pending, active, or cancelled.

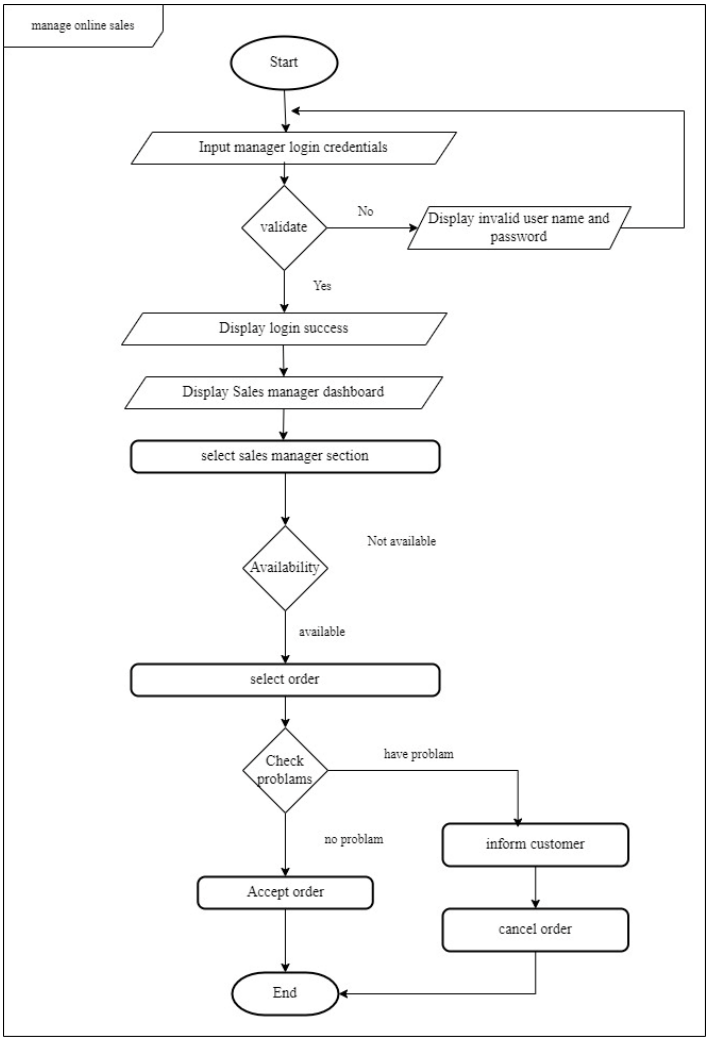
4. **Implement Credit Card Selection in Checkout**:
   - Add functionality during checkout that allows customers to use previously saved credit cards.

7.7.2 Innovative parts

1. Inventory Management with Real-Time Updates: The functionality to manage inventory with delete and update capabilities in real-time improves operational efficiency and reduces stock discrepancies.

2. **Dynamic Cart Management:**
   - **Real-time Availability Alerts**: Implementing real-time stock checks as users adjust quantities in the cart can provide immediate feedback, helping customers avoid frustration.
   - **Auto-Update Subtotals**: As customers adjust quantities or remove items, the cart automatically updates the subtotal and total costs without needing a page refresh.
3. **Order Summary with Customization**: When customers download the order summary, allow them to customize the format (PDF, PNG) and include optional notes or special offers for future purchases.
4. **Intelligent Search Functionality**: The search bar not only helps users find specific items quickly but could also utilize predictive text or suggestions based on popular searches or user history, enhancing the browsing experience.

7.7.3 Flow Chart



7.7.4 Algorithms Used

Filter items based on the search term

```
const filteredItems = items.filter(item =>
item.name.toLowerCase().includes(searchTerm.toLowerCase())
);
return (
<div className="flex">
<ManagerHeader />
<div className="container mx-auto mt-8 px-4">
<div className="flex justify-between mb-6">
<h2 className="text-6xl font-semibold">Inventory Items</h2>
<input
type="text"
placeholder="Search by item name"
className="border border-gray-300 rounded-lg px-4 py-2 text-xl"
value={searchTerm}
onChange={(e) => setSearchTerm(e.target.value)}
/>
</div>
```

Report generation

```
const generateReport = (orderData) => {
const doc = new jsPDF();
doc.text('Order Report', 14, 20);
// Customer Info
doc.text('Customer Information:', 14, 30);
```

```
doc.text(`Name: ${orderData.customerInfo.name}`, 14, 36);

doc.text(`Address: ${orderData.customerInfo.address}`, 14, 42);

doc.text(`Phone: ${orderData.customerInfo.phone}`, 14, 48);

doc.text(`Email: ${orderData.customerInfo.email}`, 14, 54);
```

Order Summary

```
doc.text('Order Summary:', 14, 92);

doc.autoTable({

startY: 98,

head: [['Item', 'Quantity', 'Price']],

body: orderData.items.map(item => [item.name, item.quantity, `LKR
${item.price.toFixed(2)}`])

});

doc.text(`Total Amount: LKR ${orderData.totalAmount}`, 14, doc.autoTable.previous.finalY
+ 10);

doc.save(`order_${Date.now()}.pdf`);

};
```

7.7.5 Pseudo Code

Filter items based on the search term

```
BEGIN
  // Convert search term to lowercase for case-insensitive search
  SET lowerCaseSearchTerm = convertToLowerCase(searchTerm)

  // Filter items by checking if each item's name includes the search term
  FOR each item in items
    SET lowerCaseItemName = convertToLowerCase(item.name)
    IF lowerCaseItemName contains lowerCaseSearchTerm THEN
      ADD item to filteredItems
    ENDIF
  ENDFOR

  // Return filtered items
```

```
  RETURN filteredItems

  // Render the filtered list and search input
  DISPLAY ManagerHeader
  BEGIN container
    DISPLAY "Inventory Items" as heading
    DISPLAY text input for searchTerm
    ON text input change:
      SET searchTerm = event.target.value
      CALL filterItems()
  END container
END
```

Report generation

```
BEGIN
  // Create a new PDF document
  INITIALIZE doc as jsPDF document

  // Add report title
  CALL doc.text with 'Order Report' at position (14, 20)

  // Add customer information to the document
  CALL doc.text with 'Customer Information:' at position (14, 30)
  CALL doc.text with 'Name: ' + orderData.customerInfo.name at position (14, 36)
  CALL doc.text with 'Address: ' + orderData.customerInfo.address at position (14, 42)
  CALL doc.text with 'Phone: ' + orderData.customerInfo.phone at position (14, 48)
  CALL doc.text with 'Email: ' + orderData.customerInfo.email at position (14, 54)

  // Add order summary heading
  CALL doc.text with 'Order Summary:' at position (14, 92)

  // Generate table with order details (item name, quantity, price)
  INITIALIZE tableData as empty array
  FOR each item in orderData.items
    SET row = [item.name, item.quantity, formatPrice(item.price)]
    ADD row to tableData
  ENDFOR

  // Add table to the PDF document
  CALL doc.autoTable with headers ['Item', 'Quantity', 'Price'] and body as tableData starting
at position (98)
```

```
  // Add total amount to the report
  SET totalYPosition = doc.autoTable.previous.finalY + 10
  CALL doc.text with 'Total Amount: LKR ' + orderData.totalAmount at position (14, totalYPosition)

  // Save the PDF file with a unique filename
  CALL doc.save with filename 'order_' + getCurrentTimestamp() + '.pdf'
END
```

7.8 Vehicle Management – IT22279798

7.8.1 Completion level of the features

Test ID	Test Case	Step	Description	Expected Result	Actual Result	Completed (Yes/No)
1	Manage vehicle accounts	1.1	Enter vehicle details (make, model, year, VIN, license plate, and maintenance schedule)	Vehicle successfully added to the system, including maintenance schedule, and stored in the centralized database	Entered details successfully	yes
		1.2	View Vehicle Details	Vehicle details (make, model, year, VIN, license plate, maintenance status) are displayed correctly	Get all the vehicle details into All vehicle page.	**ye**s
		1.3	Update Vehicle Details	Vehicle details updated successfully and reflected in the centralized database, including updated maintenance info	Show all updated details in the vehicle page	yes

		1.4	Delete Vehicle	Vehicle successfully removed from the system and no longer exists in the centralized database	Successfully delete vehicle from the system.	yes
2	Validate Vehicle Exists	2.1	Attempt to add a vehicle with a duplicate VIN	System should reject the duplicate entry and display a validation error message indicating VIN exists	Successfully reject the duplicate entry to system.	yes
3	Check Maintenance Time Left	3.1	View maintenance schedule and time left for next maintenance (e.g., 2 years remaining from last maintenance)	System correctly calculates and displays the time left for the next maintenance based on the predefined interval	Successfully calculate and displays the time.	yes
4	Update Maintenance Status	4.1	Record completion of a maintenance cycle and reset maintenance time	System updates the last maintenance date and recalculates the time remaining for the next maintenance cycle	Successfully update the last and next maintenance date	yes

Completed Features:

- Vehicle registration
- View vehicle details
- Update vehicle details
- Delete vehicle details
- Duplicate vehicle validation
- Vehicle performance monitoring

Work to be completed:

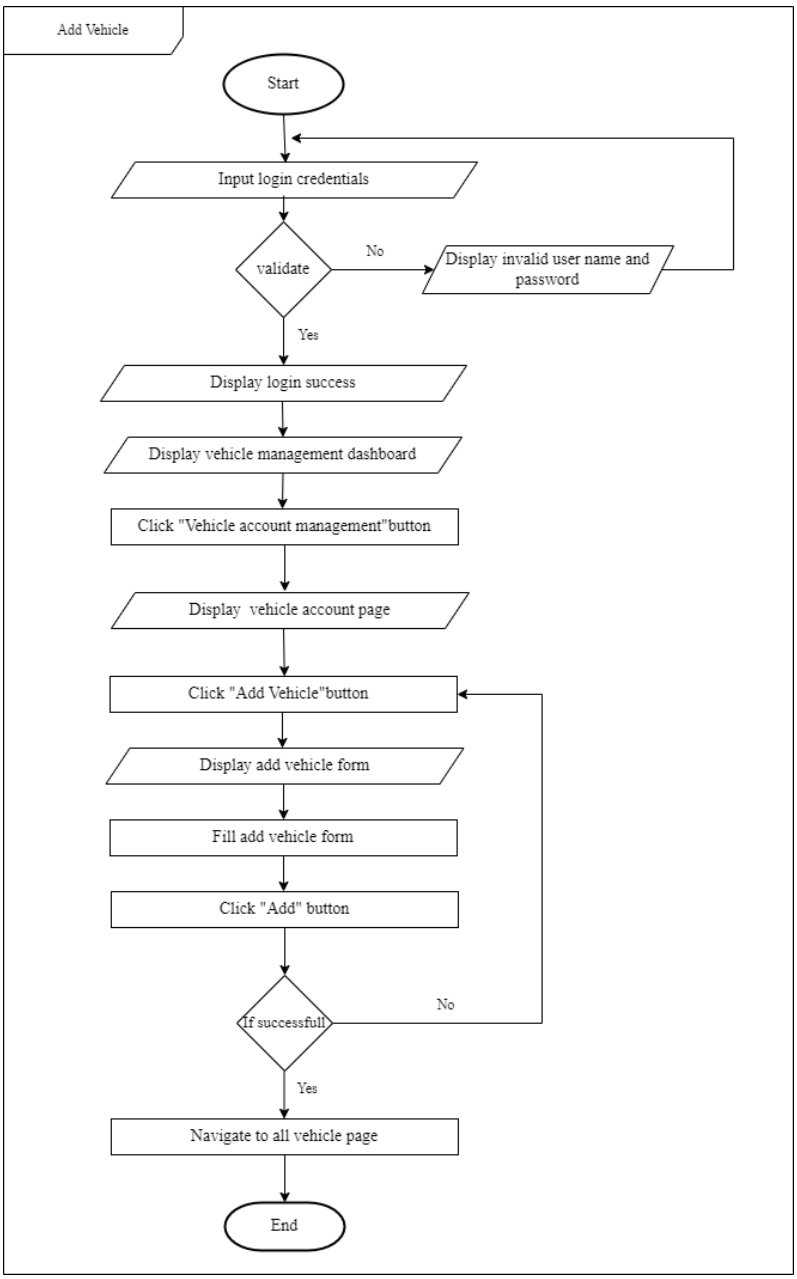
- System notification
- Maintenance tracking
-

7.8.2 Innovative parts of the project

Automated Reminder Scheduling:

- The system tracks each vehicle's last service date and sets automated reminders for the next service, which occurs every 2 years.
- The system calculates the next service date based on the last recorded service and generates a reminder well in advance.

7.8.3 Flow Chart



7.8.4 Algorithms used

Searching vehicles

```
// Function to filter vehicles based on VIN input
useEffect(() => {
  // Step 1: Check if the search input is empty
  if (searchVIN.trim() === '') {
    // Step 2: If empty, display all vehicles
    setFilteredVehicles(vehicles);
  } else {
    // Step 3: If not empty, filter vehicles by VIN
    const filtered = vehicles.filter((vehicle) =>
      vehicle.VIN.includes(searchVIN.trim())
    );

    // Step 4: Update the filtered vehicles in state
    setFilteredVehicles(filtered);
  }
}, [searchVIN, vehicles]);  // Runs whenever searchVIN or vehicles change
```

Vehicle data fetching:

```
// Function to fetch vehicle data from an API
const fetchVehicleData = async () => {
  // Step 1: Set the loading state to true
  setIsLoading(true);

  try {
    // Step 2: Make an API call to fetch vehicle data
    const response = await axios.get('http://localhost:8070/Vehicle/all');

    // Step 3: Filter vehicles by maintenance date
    const filteredVehicles = response.data.filter(
      (vehicle) => new Date(vehicle.nextMaintenanceDate) > new Date()
    );

    // Step 4: Update state with vehicle data
    setVehicles(response.data); // All vehicles
    setFilteredVehicles(filteredVehicles); // Filtered vehicles

  } catch (error) {
    // Step 5: Handle any errors during fetching
    console.error("Error fetching vehicle data:", error);
```

```
    setError(error);
  } finally {
    // Step 6: Set loading state to false after fetching
    setIsLoading(false);
  }
};

// Trigger the fetch when the component mounts
useEffect(() => {
  fetchVehicleData();
}, []);  // Runs only on the first render
```

7.8.5 Pseudo Code

```
BEGIN

  # Centralized database for vehicles
  DATABASE vehicles

  # Function to create a new vehicle entry
  FUNCTION createVehicle(make, model, year, VIN, license_plate, maintenance_interval):
    IF VIN or license_plate EXISTS in vehicles:
      PRINT "Error: Duplicate vehicle entry"
      RETURN
    ELSE:
      last_maintenance_date = CURRENT_DATE
      next_maintenance_date = last_maintenance_date + maintenance_interval
      STORE (make, model, year, VIN, license_plate, last_maintenance_date,
next_maintenance_date) IN vehicles
      PRINT "Vehicle successfully registered"
    END IF
  END FUNCTION

  # Function to read vehicle details
  FUNCTION readVehicle(VIN_or_license_plate):
    vehicle = FETCH FROM vehicles WHERE VIN_or_license_plate
    IF vehicle EXISTS:
      PRINT vehicle details
      maintenance_time_left = next_maintenance_date - CURRENT_DATE
      PRINT "Time left for maintenance: " + maintenance_time_left
    ELSE:
      PRINT "Vehicle not found"
```

```
  END IF
 END FUNCTION

 # Function to update vehicle details
 FUNCTION updateVehicle(VIN_or_license_plate, updated_make, updated_model, updated_year, updated_license_plate):
  vehicle = FETCH FROM vehicles WHERE VIN_or_license_plate
  IF vehicle EXISTS:
   UPDATE vehicle WITH updated_make, updated_model, updated_year, updated_license_plate
   PRINT "Vehicle details updated successfully"
  ELSE:
   PRINT "Vehicle not found"
  END IF
 END FUNCTION

 # Function to delete a vehicle
 FUNCTION deleteVehicle(VIN_or_license_plate):
  vehicle = FETCH FROM vehicles WHERE VIN_or_license_plate
  IF vehicle EXISTS:
   DELETE FROM vehicles WHERE VIN_or_license_plate
   PRINT "Vehicle successfully deleted"
  ELSE:
   PRINT "Vehicle not found"
  END IF
 END FUNCTION

 # Function to calculate time left for maintenance
 FUNCTION calculateMaintenanceTime(last_maintenance_date, maintenance_interval):
  next_maintenance_date = last_maintenance_date + maintenance_interval
  time_left = next_maintenance_date - CURRENT_DATE
  RETURN time_left
 END FUNCTION

END
```