Tab 1

Table of Contents

1.Project Charter

1.1 **Purpose and objective:**

The purpose of this project is to develop an integrated Point-of-Sale (POS) system tailored to meet the operational needs of retail stores.The system aims to facilitate easy transactions, improve inventory management, and provide cutting-edge analytics for enhancing decision-making. By centralizing the data from retail stores, warehouses, and headquarters, POS software will allow seamless coordination, improved operational efficiency, and foster long-term business growth for the client.

Key Objectives:

- Deliver a reliable yet user-friendly point-of-sale (POS) software application for transaction processing, inventory management, and customer engagement.
- Allow online and real-time tracking and synchronization of inventory between retail stores, warehouses, and headquarters functions.
- Provide advanced reporting and analytics tools to support data-driven decision-making at the headquarters level.
- Ensure compatibility with a wide range of retail environments and scalability to support future business growth.
- Facilitate a smooth transition to the new system through comprehensive user training, documentation, and post-deployment support.

1.2 Project Justification:

POS will provide features to retailers to streamline their operations, also reduction in processing time, and improve customer experiences by centralizing transaction and inventory data. By integrating with real-time data access and integrating with existing databases, the POS software will have:

- Faster, and more accurate sales processing.
- Inventory management tools for real-time stock monitoring.
- Analytics for the sales and reporting for HQ to optimize operations.

Our software KPI will be configurable to support various store environments and it’s scalable to meet the future business growth.

1.3 Project Scope and Structure:

Project Scope Management:

The POS software will consist of three primary components:

- Retail POS Application: Optimized User interface for transaction processing, customer management, store Inventory management, and loyalty program support.
- Warehouse Inventory: Real-time inventory management to synchronize stock across all retail locations and the warehouse itself.
- HQ Analytics and Reporting: HQ can generate a wide variety of reports to get insights into the sales from different retail stores. It is responsible for transporting the stocks from warehouses to the retail stores.

The Work Breakdown Structure (WBS) for this project includes requirements gathering, system design, software development, testing, deployment, and client training sessions. To make sure that the project team does not get distracted, and remains more focused on software delivery, our team will be collaborating with the client to define deliverables explicitly related to software performance and integration.

Deliverables/Features :

1. POS software application for transaction processing, inventory, and reporting.
2. Integrated database system that keeps track of real-time inventory and sales synchronization.
3. Comprehensive and in-depth reporting tools for HQ data analysis and decision-making.
4. Detailed user documentation and training sessions for client team members.
5. Post-deployment support for software maintenance and issue resolution.

1.4 Stakeholder Management

Key Stakeholders

For the project, these are the following stakeholders, each stakeholder has unique needs and requirements for the system.

- Warehouse Staff - Require precise estimates, real-time inventory data for accurate stock management.
- Retail Store Managers – Need a user for inputting, uploading, and processing data for analytics.

- IT Department – Supervise integration and maintenance of the POS software within the client's existing system.
- HQ Management – uses analytics and reporting tools to monitor different sales patterns and create strategic choices.
- Client companies – Oversee the project, provide funding, and approve final deliverables to ensure the POS system aligns with their business goals.
- End Customers – Indirectly impacted by the system; benefit from faster transactions and an improved shopping experience.
- Store Employees – Operate the POS system for daily transactions and inventory updates; require intuitive and efficient software.

Stakeholder Engagement Plan

Clear communication, requirement validation and closed feedback loops will be the main forte of our stakeholder management strategy.

1. Weekly Updates – Stakeholders will receive weekly updates during the process of development and provide feedback.
2. Requirements Reviews – Regular sessions with stakeholders to ensure the features of the product meet their demands
3. Direct Communication Channels – To facilitate communication with the stakeholders, Slack and Microsoft Teams will be used and each stakeholder will have a designated point of contact to quickly handle any issue or concerns.

1.5 Resource and Procurement Management

Our only responsibility in procurement as a software supplier is to select and manage the tools for software development, testing, and integration. All physical point-of-sale hardware and associated accessories will be the client's responsibility. To enable seamless client setup, our staff will guarantee software compatibility with widely used POS hardware variants.

Internal Resources

1. Development Team – Responsible for POS software configuration, testing and coding.
2. Client Support Team – To help client's IT team with any integration issues, the client support team provides post deployment support.

1.6 Risk Management

Risk Management Plan:

Since the POS software is indispensable for daily operations in retail stores, several risks are predicted such as integration challenges of the software with several types of databases, data flow issues and human error whilst using the software. Our risk management plan includes:

1. **Compatibility Testing**: The software is going to be tested beforehand with different types of databases and versatility will be ensured across various databases.
2. **Data Synchronization**: Frequent system audits and comparison of database with transaction logs to ensure smooth and error-prone data transfer among retail stores, warehouses, and HQ.
3. **User Adaptation**: Thorough training for different types of end user's for adapting to the new software to reduce potential human errors.

1.7 Timeline and Scheduling

Project Phases and Milestones:

1. **Design and Planning** (*Month 1*): Gatherings of needs and requirements, designing software and getting approval.
2. **Development and Integration** (*Months 2-4*): Development and initial integration of the software, unit and integration testing for Quality Assurance.
3. **Testing and Quality Assurance (QA)** (*Month 5*): Different testing phases, including user acceptance testing and system testing with feedback from clients.
4. **Deployment and Training** (*Month 6*): Final deployment of the software, end users training and post-deployment maintenance.

Progress tracking will be managed via JIRA, where requirements and iterations will be recorded. Git will be used for providing a collaborative environment for developers, continuous integration and deployment (CI/CD), automated build and test processes.

1.8 Quality Assurance Plan

To provide a solid foundation for the deployment of quality assurance procedures, complete validation strategies that focus on the structured evaluation of all system components and stakeholders' satisfaction indicators would be followed. Verification Protocol Hierarchical test methodologies are as follows:

1. **Unit Testing**: Each software module will go through an in-depth testing to guarantee functional reliability.

2. **System Integration Testing**: Cross-checking for compatibility with the client's old databases.
3. **User Acceptance Testing (UAT)**: Required to ensure the software meets the user's workflow requirements and standards of usability, UAT will involve company's personnel.

Testing Tools	
GitHub Actions	Continuous integration and testing using automated CI/CD.
TDD	Developers will use Test Driven Development approach to make sure the system is working beforehand and there are less chances of bugs in the system.
JIRA	Used to keep track of problems found during testing and guarantee prompt fixes.

1.9 Communications and Collaboration Tools

Since meeting the stakeholder's expectations and completing the project on schedule depend on how effective the communication between the internal team and the client. Following resources will be incorporated into our communication strategy:

1. **Slack**: The platform for conversations and real-time updates about projects.
2. **Zoom and Microsoft Teams**: For online conferences and weekly stakeholder progress reports.
3. **JIRA**: To monitor sprints, requirements, and project progress.
4. **GitHub:** For software deployment management, CI/CD, and code versioning.

1.10 **Project close-out and acceptance criteria**

Acceptance Criteria

The POS software project will be considered complete when:

1. The system satisfies all functional criteria, according to the client's UAT validation.
2. The client's employees receive training and documentation to ensure a seamless system transition.

3. The program provides precise transaction and inventory data by integrating easily with the client's systems.

1.11 Project close-out

Upon the project completion, a project review meeting will be organized with all key stakeholders involved to get input and make sure that the project fulfills their needs and record any possible areas for future development. For a smooth transition, post-deployment assistance will be offered for a maximum of three months to handle minor modifications or software-specific problems.

The goal of this POS software project is to provide our client's retail operations with an effective, integrated solution that improves inventory control, transaction processing, and data-driven decision-making. By emphasizing software compatibility, quality control, and stakeholder cooperation, we hope to provide our customer with a point-of-sale system that satisfies present operational requirements and facilitates scalable expansion.

2. Project Scope

Project Title: Development of an Integrated Point-of-Sale (POS) System for Retail Stores

Overview

The purpose of this project is to develop an integrated Point-of-Sale (POS) system tailored to meet the operational needs of retail stores. The system aims to facilitate easy transactions, improve inventory management, and provide cutting-edge analytics for enhancing decision-making. By centralizing the data from retail stores, warehouses, and headquarters, POS software will allow seamless coordination, improved operational efficiency, and foster long-term business growth for the client.

Scope Statement:

The project will include the development, integration, deployment and support for an integrated POS software system. The software will include integration of retail store operations, warehouse inventory synchronization and headquarters analytics.

Deliverables /Features :

1. **POS software application for transaction processing, inventory, and reporting.**

   The software will provide a platform for efficient transaction processing, including sles, refunds and payment methods. It will also include features for managing and tracking the inventory level and help with operational reports.

2. **Integrated database system that keeps track of real-time inventory and sales synchronization.**

   With an integrated database, the sales and inventory information will be synchronized in real-time and accessible to retail stores, warehouses, and the head office. This grows to an accurate stock count, minimizing differences, and triggering automated workflows for restocking alerts and order fulfillment.

3. **Comprehensive and in-depth reporting tools for HQ data analysis and decision-making.**

   Notable implementations will provide headquarters insight into sales trends, inventory status, and store performance. Customizable dashboards along with data visualization options will make strategic planning and operational optimization data-driven.

4. **Detailed user documentation and training sessions for client team members.**

   In order to ensure smooth adoption of the POS system, comprehensive user manuals, quick-start guides, and video tutorials will be rolled out. Training sessions tailored for retail staff, warehouse personnel, and IT teams will focus on system functionality and best practices.

5. **Post-deployment support for software maintenance and issue resolution.**

   There will be a dedicated support phase for three months following deployment, to help troubleshoot problems, update software, and make minor modifications. This ensures that the system works robustly while also responding to initial user feedback.

Scope Exclusions:

1. Procuring or setting up physical hardware like POS terminals and printers.
2. Developing features not included in the approved backlog.
3. Integrating systems not identified during initial planning.

Constraints and Assumptions

1. The project will follow a six-month timeline with 2-week sprints.
2. Stakeholder feedback will guide feature prioritization and refinements.
3. The client will ensure access to infrastructure and databases to facilitate development and integration.
4. All features will comply with relevant data security and privacy regulations.

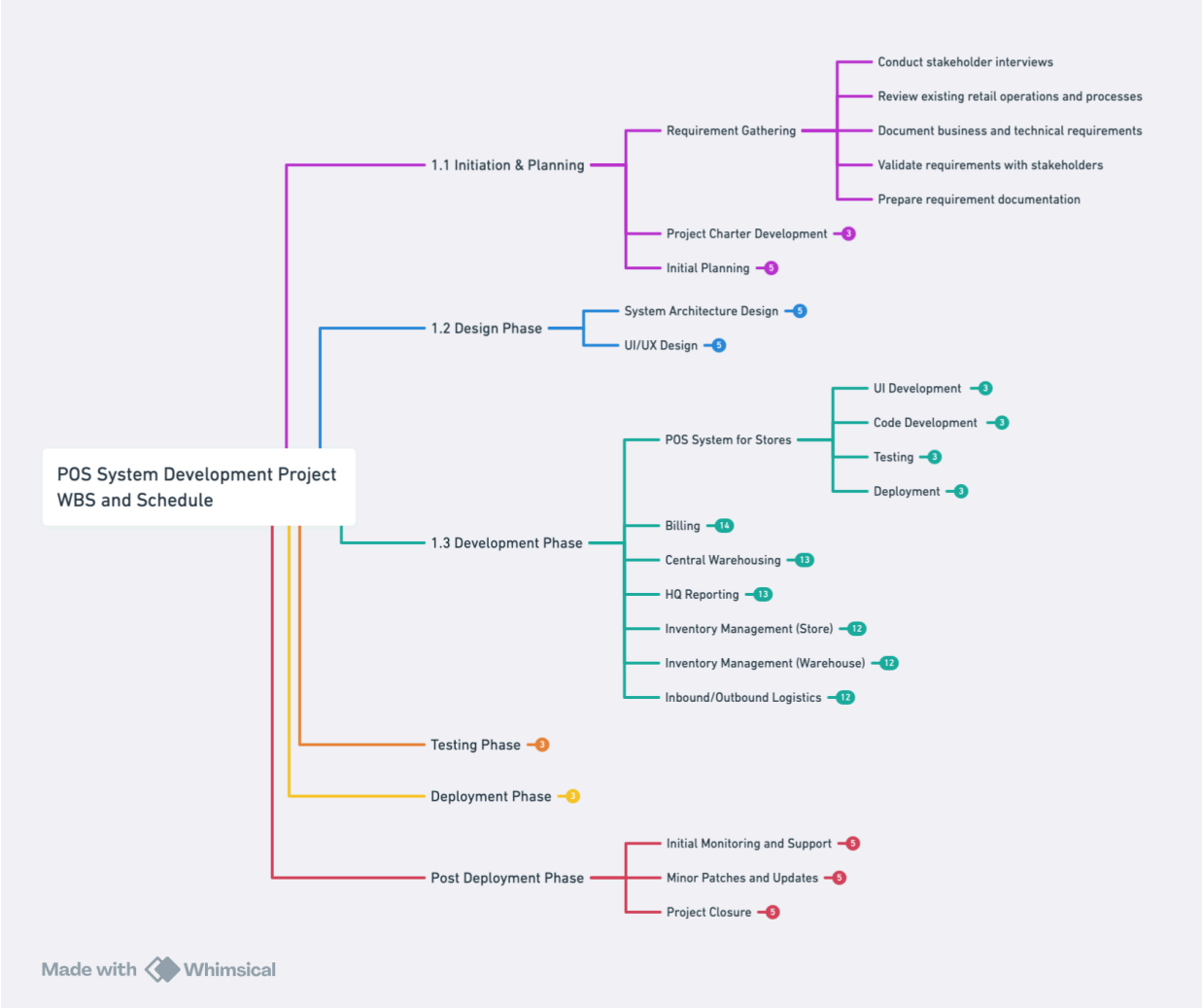
Milestones

1. **UI Development Completion**: All user interfaces will be designed and finalized by February 26.
2. **Code Development Completion**: The coding for all features will be completed by March 4.
3. **Feature Testing Completion**: Testing of individual features will be finalized by March 10.
4. **Integration Testing Completion**: Integration testing for all features will conclude by March 13.
5. **Deployment**: All features will be deployed between March 25 and March 29.

Detailed milestone table Feature wise.

Feature	UI Development	Code Development	Testing	Integration Testing	Deployment
POS System for Stores	Jan 2-Jan 8	Jan 9-Jan 15	Jan 16-Jan 20	Jan 21-Jan 23	Mar 25-Mar 29
Billing	Jan 9-Jan 15	Jan 16-Jan 22	Jan 23-Jan 27	Jan 28-Jan 30	Mar 25-Mar 29
Central Warehousing	Jan 16-Jan 22	Jan 23-Jan 29	Jan 30-Feb 3	Feb 4-Feb 6	Mar 25-Mar 29
HQ Reporting	Jan 30-Feb 5	Feb 6-Feb 12	Feb 13-Feb 17	Feb 18-Feb 20	Mar 25-Mar 29
Inventory Management (Store)	Feb 6-Feb 12	Feb 13-Feb 19	Feb 20-Feb 24	Feb 25-Feb 27	Mar 25-Mar 29
Inventory Management (WH)	Feb 13-Feb 19	Feb 20-Feb 26	Feb 27-Mar 3	Mar 4-Mar 6	Mar 25-Mar 29
Inbound/Outbound Logistics	Feb 20-Feb 26	Feb 27-Mar 4	Mar 5-Mar 10	Mar 11-Mar 13	Mar 25-Mar 29

Work-Breakdown-Structure (WBS).



There are 4 levels in this project's Work breakdown structure .

They are as follows.

POS System Development Project

- 1.1 Initiation & Planning
  - Requirement Gathering
    - Conduct stakeholder interviews
    - Review existing retail operations and processes
    - Document business and technical requirements
    - Validate requirements with stakeholders
    - Prepare requirement documentation
  - Project Charter Development
    - Draft initial project charter
    - Review project charter with key stakeholders
    - Finalize and approve the project charter
  - Initial Planning
    - Define project scope
    - Set high-level project timeline and budget
    - Identify project milestones
    - Develop a risk management strategy
    - Assemble project team and assign roles
- 1.2 Design Phase
  - System Architecture Design
    - Develop overall architecture blueprint
    - Design data flow diagrams
    - Create database schema
    - Validate architecture with IT department

- Document architecture specifications
- UI/UX Design
  - Create initial wireframes and prototypes
  - Collect user feedback on prototypes
  - Make revisions based on feedback
  - Finalize detailed UI/UX designs
  - Prepare design documentation
- 1.3 Development Phase
  - POS System for Stores
    - UI Development
      - Code the main POS user interface
      - Implement user-friendly navigation
      - Ensure responsive design across devices
    - Code Development
      - Develop transaction processing functionality
      - Implement customer management logic
      - Integrate with database for real-time updates
    - Testing
      - Conduct unit testing for POS functionality
      - Fix bugs and retest
      - Validate responsiveness on multiple devices
    - Deployment
      - Finalize deployment package
      - Set up live POS system
      - Conduct operational checks
  - Billing
    - UI Development

        - Design billing interface with GST and tax fields
        - Develop payment gateway integration UI
    - Code Development
        - Implement backend billing logic
        - Develop transaction history tracking
        - Integrate loyalty programs
    - Testing
        - Test accuracy of bill calculations
        - Validate integration with inventory and POS
        - Fix bugs and optimize performance
    - Deployment
        - Deploy billing module with POS system
        - Verify data synchronization between modules
- Central Warehousing
    - UI Development
        - Design stock overview interface
        - Implement inbound/outbound logistics UI
    - Code Development
        - Develop logic for stock movement tracking
        - Integrate warehouse module with inventory
    - Testing
        - Validate stock updates in real time
        - Conduct load testing for large data sets
        - Fix issues and retest
    - Deployment
        - Roll out central warehousing module
        - Verify system connectivity

- HQ Reporting
  - UI Development
    - Design reporting dashboards
    - Implement charts and data visualization
  - Code Development
    - Build analytics backend for reporting
    - Enable data aggregation from all modules
  - Testing
    - Validate accuracy of reports
    - Optimize dashboard performance
    - Fix bugs and retest
  - Deployment
    - Deploy reporting module
    - Provide user training on report usage
- Inventory Management (Store)
  - UI Development)
    - Design store-level inventory interface
    - Develop stock adjustment UI
  - Code Development)
    - Implement inventory management logic
    - Develop integration with POS system
  - Testing
    - Validate stock updates and adjustments
    - Fix issues and optimize logic
  - Deployment
    - Deploy store inventory module
    - Conduct final operational checks

- Inventory Management (Warehouse)
  - UI Development
    - Create interface for warehouse-level inventory
    - Develop transfer and restocking features
  - Code Development
    - Implement logic for inter-warehouse transfers
    - Optimize for bulk inventory updates
  - Testing
    - Test bulk inventory processes
    - Validate data synchronization
  - Deployment
    - Deploy warehouse inventory module
    - Verify system connectivity
- Inbound/Outbound Logistics
  - UI Development
    - Create inbound/outbound tracking interface
    - Design real-time vehicle tracking UI
  - Code Development
    - Implement logic for shipment tracking
    - Integrate with central warehousing system
  - Testing
    - Validate shipment tracking accuracy
    - Conduct performance testing
  - Deployment
    - Deploy logistics module
    - Ensure operational readiness
- Testing Phase

    - Conduct thorough unit testing for all modules
    - Perform performance and load testing
    - Validate end-to-end workflows in UAT sessions
- Deployment Phase
    - Consolidate all modules for deployment
    - Conduct final operational checks
    - Provide training and documentation for users
- Post Deployment Phase
    - Initial Monitoring and Support
        - Monitor system performance for the first few weeks
        - Address user-reported issues quickly
        - Provide ongoing support as needed
        - Collect usage data for optimization
        - Report findings to stakeholders
    - Minor Patches and Updates
        - Identify required patches from feedback
        - Develop and test patch updates
        - Deploy patches to client sites
        - Validate updates and fixes
        - Document all patch activities
    - Project Closure
        - Conduct post-deployment review meeting
        - Collect and document lessons learned
        - Obtain final stakeholder sign-off
        - Archive project documents and codebase
        - Close out project and transition to maintenance phase

Scope Verification Procedures

1. Formats Due in Relation to the Scope Statement
Procedure: Each time you finish a deliverable compare it to the project scope statement and the Work Breakdown Structure (WBS). Make certain all abilities (such as initiating and processing transactions, managing inventory, and analysis) listed are to be contained.
Responsible Party: Project Manager, Quality Assurance, Tester or the Client.
Documentation: The following are the scope checklist and the deliverable acceptance form.

2. Conduct Stakeholder Review
Procedure: Schedule official feedback meetings with the target audiences such as, Warehouse Staff, Retail Store Managers, IT Department, HQ Management, and Client Representatives to ensure that objectives are achieved as expected.
Frequency: They should be tested at the end of each of the milestones before the system is deployed to the users.
Responsible Party: The Project Manager and Development Team bring good news for both companies and employees throughout this crisis situation.
Tools: Exhibit 3, simulations, functional models, and test beds.

3. Conduct User Acceptance Testing,UAT
Procedure: Solicit the Store Employees, IT Department or any other target end-user which is to conduct UAT as outlined in the test script. Formal and informal feedback to guarantee that the system is usable, structural, and responsive to a range of requirements of the different stakeholder categories.
Frequency: During the UAT phase which is on the month 5.
Responsible Party: QA Team, and the representatives from the stakeholders.
Documentation: Test, reports UAT, Test, reports User, Feedback, Forms.

4. Check Integration and Real-time Sync
Procedure: verify actual time refresh of inventory status on POS against current databases and systems of the retail stores, warehouses and headquarters . Check compatibility and scalability.
Frequency: In system testing and integration testing stage.
Responsible Party: Development Team and IT Department both are same.
Tools: Depending on the outcome of the testing, the performance testing tools and the integration logs used may change.

5. Review Quality Metrics
Procedure: Measure how well a business is doing against predetermined quality standards like system availability, transaction through put time, inventory update time and disability access.

Frequency: In testing and before going live.
Responsible Party: QA Team.
Tools: Software tracking tools (for example JIRA, GitHub Actions).

6. Get the Approval of the Formal Stakeholder
Procedure: Obtain sign off, on paper, from all the stakeholders, to signal their acceptance of the output. Clear any remaining issues or minor tweaking up before the roll out.
Frequency: During the UAT, after and at the conclusion of the final inspections.
Responsible Party: Thus, Project Manager and Client Representatives.
Documentation: Sign-Off Form.

7. Training and documentation
Procedure: Guarantee that user training programs are well delivered and well documented (orientation, manuals, quick start instructions, technical information sheets).
Frequency: Before deployment (Month 6).
Responsible Party: Training Team and Client Representatives.
Documentation: Training feedback report & Training feedback documentation validation forms.

8. Support after the Deployment of a New Product
Procedure: Continued support after the shift to the new system to solve any problems that may have escaped the post-implementation assessment. Keep notes for future improvement of the records.
Frequency: This is in the course of the first 3 months after the products have been deployed into the market.
Responsible Party: Support Team.
Tools: Related to these are the issue tracking tools and the feedback logs.

Acceptance Criteria
The project deliverables will be accepted when:

1. All features are checked against the scope statement and have functional requirements only.
2. Getting approval from the stakeholders after efficient completion of UAT process.
3. Training is carried out, and all the necessary papers are produced.
4. According to the remaining measurable quality attributes (e.g., uptime, transaction time), all are achieved.
5. Managers, directors and shareholders put their signature to the delivered outcomes.

Deliverables Checklist

Deliverable	Verification Method	Status

Retail POS Application	UAT, stakeholder review	Pending
Warehouse Inventory Management Module	Integration testing, UAT	Pending
HQ Analytics and Reporting Tools	Stakeholder review, quality metrics	Pending
Documentation and Training	Stakeholder review, quality metrics	Pending
Post-Deployment Support Plan	Feedback tracking	Pending

3.Schedule Management

The project follows a six-month timeline with bi-weekly sprints. Each sprint focuses on delivering specific increments of functionality, as identified in the Work Breakdown Structure. Planning includes:

- Breaking down tasks into measurable function points.
- Prioritizing tasks based on stakeholder feedback and business value.
- Assigning tasks to team members with clear deadlines

Schedule Development Methodology

The project employs an Agile-SCRUM methodology with two-week sprints, which according to Schwaber and Sutherland optimizes predictability and risk control.[6] The project is structured across six key phases, aligning with contemporary software development practices that emphasize iterative development and continuous integration.[7]

The schedule is structured around the following key milestones.

Month 1: Requirements gathering, system design, and initial planning.

Months 2-4: Development and integration, including unit testing and sprint reviews.

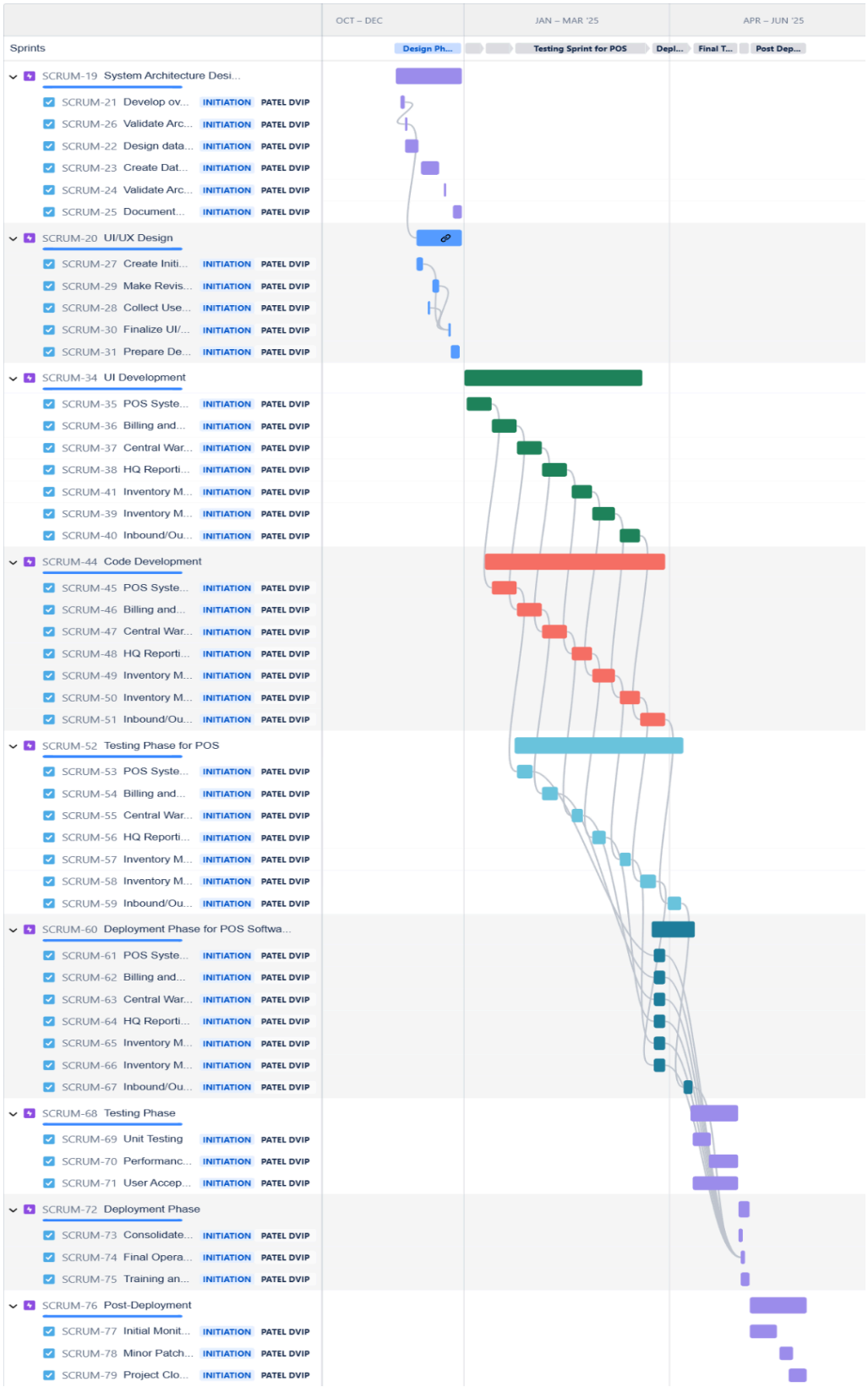
Month 5: Comprehensive testing, including User Acceptance Testing (UAT).

Month 6: Deployment, training, and post-deployment support .

Each sprint delivers fully functional modules, tracked via JIRA. Dependencies are managed using Gantt charts, and progress is monitored through daily stand-ups and weekly reviews.

Project Timeline

The timeline structure follows the proven phase-gate approach recommended by Cooper (2019), with clear transitions between major project phases.[8]



Project timeline developed with JIRA

Tools and Techniques

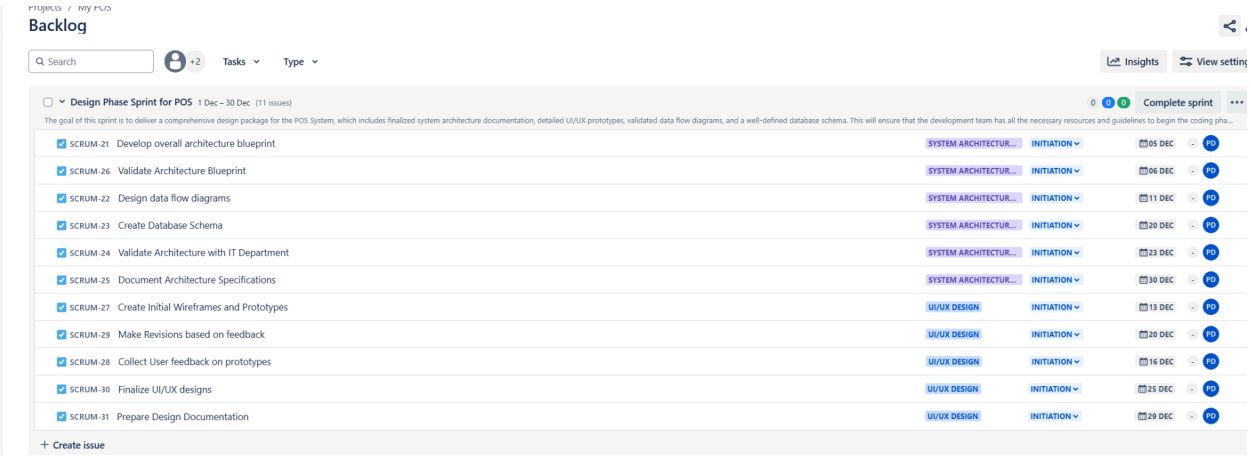
- **JIRA:** Tracks tasks, sprint progress, and overall timeline adherence.
- **GitHub:** Ensures continuous integration and delivery (CI/CD).
- **Slack and Microsoft Teams:** Facilitate communication for resolving schedule conflicts and monitoring progress.
- **Gantt Charts:** Visualize dependencies and critical paths in the schedule.

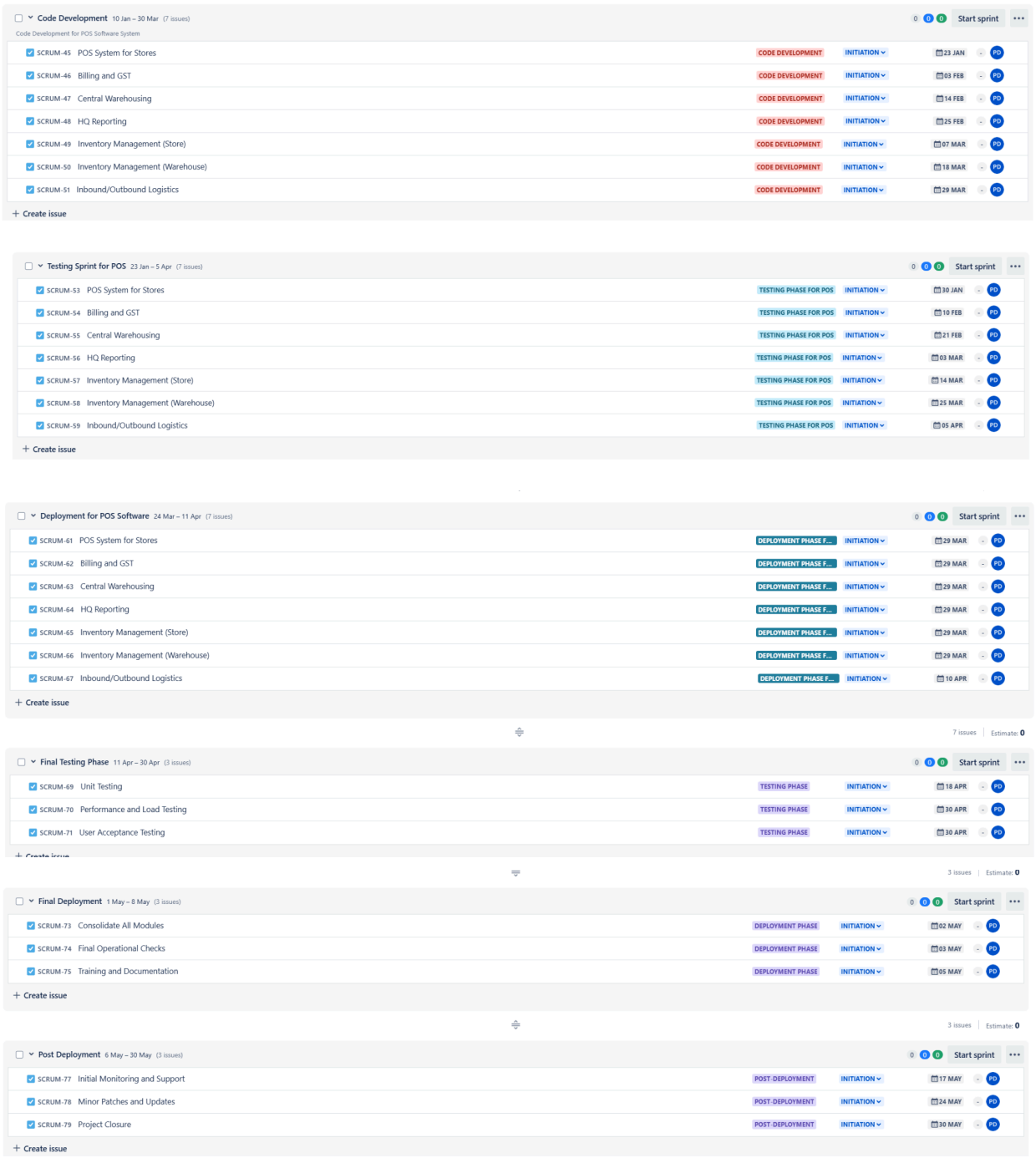
Sprint Planning and Control

Sprint management follows the Scaled Agile Framework® (SAFe®) principles (Leffingwell, 2020), incorporating daily stand-ups and regular retrospectives to maintain project velocity.

Backlog Visualization

To support effective sprint planning and task prioritization, the following backlog visualization provides a detailed breakdown of the tasks organized by phase and status. This table outlines the critical deliverables for the final deployment and post-deployment phases, offering a clear view of the progression and dependencies. It is a crucial reference for monitoring and ensuring the timely completion of all project activities.





Schedule Baselines

The schedule is built on specific milestones and deliverables, structured to maintain project cadence and quality. The following baselines are established to monitor and manage schedule performance effectively:

- **Requirements Baseline**: Completion of requirements gathering and initial planning by the end of Month 1.
- **Development Baseline**: All core development and integration tasks, including sprint reviews, to be completed by the end of Month 4.
- **Testing Baseline**: Comprehensive testing, including User Acceptance Testing (UAT), completed by Month 5.
- **Deployment Baseline**: Successful deployment and training completed, with post-deployment support initiated by the end of Month 6.

This schedule management plan provides for an iterative development process that welcomes assessments and feedback from stakeholders. The application of tried-and-true methodologies, tools, and baselines gives a clear system of how progress can be monitored and risks prevented, upon which success of the project rests.

4.Cost Management

The cost estimation plan outlines the financial and resource requirements for the successful completion of POS System Development Project. This document contains Function Point Analysis (FPA). It's a standardized metric for estimating work efforts, and to determine project complexity, resource allocation, and total costs. Additionally, it provides a breakdown of developer hours, tools, infrastructure and post-deployment support.

1. **Function Point Breakdown:**

   Function points are assigned based on the complexity and effort required for individual tasks.

   1. **Design Phase**

      1.1. System Architecture Design

Task	Complexity	Function Points (FP)
Develop Overall architecture blueprint	Medium	2
Design data flow diagrams	Medium	2
Create database schema	Medium	2
Validate Architecture with IT Department	Low	1
Document architecture specification	Low	1
Subtotal for System Architecture Design	-	8 FP

1.2 UI/UX Design

Task	Complexity	Function Points
Create initial wireframes and prototypes	High	3
Collect User feedback on prototypes	Medium	2
Make revisions based on feedback	Medium	2
Finalize detailed UI/UX designs	High	3
Prepare design documentation	Low	1
Subtotal for UI/UX Design	-	11 FP

1.3 Development Phase

1.3.1 POS System for Stores

Task	Complexity	Function Points (FP)
UI Development		
Code the main POS user interface	High	3
Implement user-friendly navigation	Medium	1
Ensure responsive design across devices	Medium	1
Subtotal for UI Development	-	**5 FP**

Code Development		
Develop transaction processing functionality	High	3
Implement customer management logic	Medium	2
Integrate with database for real-time updates	Medium	1
Subtotal for Code Development	-	**6 FP**
Testing		
Conduct unit testing for POS functionality	Medium	2
Fix bugs and retest	Low	1
Validate responsiveness on multiple devices	Low	1
Subtotal for Testing	-	**4 FP**
Deployment		
Finalize deployment package	Low	1
Set up live POS system	Medium	1
Conduct operational checks	Low	1
Subtotal for Deployment	-	**3 FP**

POS System Total: 18 FP

1.3.2 Billing

Task	Complexity	Function Points (FP)
UI Development		
Design billing interface with GST and tax fields	High	3
Develop payment gateway integration UI	Medium	2
Subtotal for UI Development	-	**5 FP**
Code Development		
Implement backend billing logic	High	3
Develop transaction history tracking	Medium	2
Integrate loyalty programs	Medium	2
Subtotal for Code Development	-	**7 FP**
Testing		
Test accuracy of bill calculations	Medium	2
Validate integration with inventory and POS	Medium	2
Fix bugs and optimize performance	Low	1
Subtotal for Testing	-	**5 FP**

Deployment		
Deploy billing module with POS system	Low	1
Verify data synchronization between modules	Medium	1
Subtotal for Deployment	-	**2 FP**

Billing Total: 19 FP

1.3.3 Central Warehousing

Task	Complexity	Function Points (FP)
UI Development		
Design stock overview interface	Medium	2
Implement inbound/outbound logistics UI	Medium	2
Subtotal for UI Development	-	**4 FP**
Code Development		
Develop logic for stock movement tracking	High	3
Integrate warehouse module with inventory	High	3

Subtotal for Code Development	-	**6 FP**
Testing		
Validate stock updates in real time	Medium	2
Conduct load testing for large data sets	High	3
Fix issues and retest	Medium	2
Subtotal for Testing	-	**7 FP**
Deployment		
Roll out central warehousing module	Low	1
Verify system connectivity	Medium	1
Subtotal for Deployment	-	**2 FP**

Central Warehousing Total: 19 FP

Final Summary of Function Points

Phase/Module	Function Points (FP)
Design Phase	19

POS System for Stores	18
Billing	19
Central Warehousing	19
HQ Reporting	20
Inventory Management (Store)	18
Inventory Management (Warehouse)	19
Inbound/Outbound Logistics	20
Testing Phase	14
Deployment Phase	10
Post-Deployment Phase	44
Total	210 FP

Effort Distribution by Role

Role	Total Function Points	Effort (Hours)	Hourly Rate (CAD)	Total Cost (CAD)
Graphic Designers	19 FP	190 hours	$30/hour	$5,700
Frontend Developers	48 FP	480 hours	$35/hour	$16,800
Backend Developers	75 FP	750 hours	$40/hour	$30,000
QA Testers	42 FP	420 hours	$30/hour	$12,600
DevOps Engineers	26 FP	260 hours	$50/hour	$13,000
Project Manager	10% of Total Hours (8640 FP * 10 hours) = 864 hours	864 hours	$55/hour	$47,520

Role-Based Labor Costs

1. Graphic Designers[17]

- **Role:** Responsible for UI/UX design (wireframes, prototypes, feedback iterations, final designs).
- **Effort:** 19 FP × 10 hours/FP = 190 hours.
- **Hourly Rate:** $30/hour.
- **Total Cost:** 190 × 30 = $5,700.

2. Frontend Developers [9]

- **Role:** Implements UI designs into functional interfaces and ensures responsiveness.
- **Effort:** 48 FP × 10 hours/FP = 480 hours.
- **Hourly Rate:** $35/hour.
- **Total Cost: 480 × 35 =** $16,800.

3. Backend Developers [9]

- **Role:** Develops backend logic, APIs, database integration, and complex modules like reporting and inventory.
- **Effort:** 75 FP × 10 hours/FP = 750 hours.
- **Hourly Rate:** $40/hour.
- **Total Cost:** 750 × 40 = $30,000.

4. QA Testers[16]

- **Role:** Performs unit testing, performance testing, UAT, and bug fixes.
- **Effort:** 42 FP × 10 hours/FP = 420 hours.
- **Hourly Rate:** $30/hour.
- **Total Cost:** 420 × 30 = $12,600.

5. DevOps Engineers[19]

- **Role:** Manages cloud infrastructure, CI/CD pipelines, and deployment tasks.
- **Effort:** 26 FP × 10 hours/FP = 260 hours.
- **Hourly Rate:** $50/hour.
- **Total Cost:** 260 × 50 = $13,000.

6. Project Manager[18]

- **Role:** Oversees the entire project, monitors progress, manages risks, and ensures delivery.
- **Effort:** 10% of total effort (864 FP × 10 hours/FP = 8640 hours).
    - 10% of 8640 = 864 hours.
- **Hourly Rate:** $55/hour.
- **Total Cost:** 864 × 55 = $47,520.

Total Labor Costs:

Role	Total Cost (CAD)
Graphic Designers	$5,700
Frontend Developers	$16,800
Backend Developers	$30,000
QA Testers	$12,600
DevOps Engineers	$13,000

Project Manager	$47,520
Grand Total	$125,620

Software Costs

1. Jira (Project Management)[11]

- **Plan:** Standard Plan.
- **Cost:** $7.75/user/month.
- **Team Size:** 9 users (Graphic Designers, Developers, QA Testers, DevOps Engineers, Project Manager).
- **Monthly Cost:** 9 × $7.75 = **$69.75**.
- **Total Cost (6 Months):** $69.75 × 6 = **$418.50**.

2. Git Version Control (GitHub/Bitbucket)[13]

- **Plan:** Team Plan.
- **Cost:** $5/user/month.
- **Team Size:** 9 users.
- **Monthly Cost:** 9 × $5 = **$45**.
- **Total Cost (6 Months):** $45 × 6 = **$270**.

3. Figma (UI Design)[12]

- **Plan:** Professional Plan.
- **Cost:** $12/user/month.
- **Team Size:** 2 users (Graphic Designers).
- **Monthly Cost:** 2 × $12 = **$24**.
- **Total Cost (6 Months):** $24 × 6 = **$144**.

4. Slack (Team Communication)[14]

- **Plan:** Pro Plan.
- **Cost:** $7.25/user/month.
- **Team Size:** 9 users.
- **Monthly Cost:** 9 × $7.25 = **$65.25**.
- **Total Cost (6 Months):** $65.25 × 6 = **$391.50**.

5. Zoom (Meetings and Collaboration)[15]

- **Plan:** Pro Plan.

- **Cost:** $14.99/license/month.
- **Licenses:** 2 licenses (Project Manager and Team Collaboration).
- **Monthly Cost:** 2 × $14.99 = **$29.98**.
- **Total Cost (6 Months):** $29.98 × 6 = **$179.88**.

6. Adobe Creative Cloud (Design Suite for UI/UX)

- **Plan:** Individual Plan.
- **Cost:** $54.99/user/month.
- **Team Size:** 2 users (Graphic Designers).
- **Monthly Cost:** 2 × $54.99 = **$109.98**.
- **Total Cost (6 Months):** $109.98 × 6 = **$659.88**.

7. Selenium (QA Testing Automation)

- **Plan:** Cloud-based Automation Tools.
- **Cost:** $150/month (shared team license).
- **Monthly Cost: $150**.
- **Total Cost (6 Months):** $150 × 6 = **$900**.

8. AWS CloudWatch (Monitoring and Logging)

- **Plan:** Usage-based (approximation for this project).
- **Cost:** $50/month.
- **Monthly Cost: $50**.
- **Total Cost (6 Months):** $50 × 6 = **$300**.

9. Other Tools (Miscellaneous)

- **Examples:** Notion, Toggl for time tracking.
- **Cost:** Approx. $50/month.
- **Monthly Cost: $50**.
- **Total Cost (6 Months):** $50 × 6 = **$300**.

Summary of Software Costs

Tool	Monthly Cost (USD)	Total Cost (6 Months)
Jira	$69.75	$418.50
Git Version Control	$45	$270
Figma	$24	$144
Slack	$65.25	$391.50
Zoom	$29.98	$179.88
Adobe Creative Cloud	$109.98	$659.88
Selenium	$150	$900
AWS CloudWatch	$50	$300
Other Tools	$50	$300
Grand Total	**$593.96**	**$3,563.76**

Grand Total for Labor + Software

Category	Total Cost (USD)
Labor Costs	$125,620
Software Costs	$3,563.76
Final Total	$129,183.76

5.Resource Management:

The resource management plan demonstrates how human resources, tools and other resources will be allocated, tracked and optimized throughout the project lifecycle. This ensures that the project is completed within scope, budget and timeline.

1. **Resource Identification:**

Resource Type	**Details**
Human Resources	Graphic Designers, Frontend Developers, Backend Developers, QA Testers, DevOps Engineers, Project Manager
Software Tools	Jira, Git, Figma, Slack, Zoom, Adobe Creative Cloud, Selenium, AWS Cloudwatch
Infrastructure	AWS EC2 instances, RDS databases, S3 storage, CloudWatch monitoring
Budget	$129,183.76 allocated for labor and software costs

2. Human Resource Allocation:

Role	**Tasks/Responsibilities**	**Number of Resources**	**Duration (Months)**
Graphic Designers	Create wireframes, prototypes, and final UI designs for all modules	1	2
Frontend Developers	Implement UI designs, ensure responsiveness, and develop frontend	2	4

	functionality for POS, billing, and reporting		
Backend Developers	Develop server-side logic, API integrations, database schemas, and backend logic for all modules	3	6
QA Testers	Conduct unit, performance, and end-to-end testing; validate functionality and optimize system performance	2	4
DevOps Engineers	Manage cloud infrastructure, set up CI/CD pipelines, and handle deployments	1	6
Project Manager	Plan, monitor, and control project execution; manage risks and resources; report to stakeholders	1	6

3. **Resource Tracking:**
    - **Project Management Tool:** Jira for task allocation and progress tracking
    - **Time Tracking tool:** Toggl for logging hours worked
    - **Communication Tool:** Slack for team collaboration and updates

4. **RACI Matrix (Responsibility Assignment Matrix):**

    Key Roles in the project

    - **R (Responsible):** Executes the task
    - **A (Accountable):** Ultimately answerable and approves work.
    - **C (Consulted):** Provides input and feedback

- **I (Informed):** It is kept updated on the ongoing progress of the project.

Task	Project Manager	Graphic Designer	Frontend Developer	Backend Developer	QA Tester	DevOps Engineer
Define Project Scope	A	I	I	I	I	I
Create Wireframes and prototypes	I	R	C	I	I	I
Implement UI Designs	I	C	R	I	I	I
Develop backend logic	I	I	I	R	I	I
Conduct Unit testing	I	I	C	C	R	I
Manage Deployments	I	I	I	I	I	R
Monitor System Performance	A	I	I	I	R	R
Conduct Stakeholder meetings	R	I	I	I	I	I

Responsibility Assignment Matrix:

This matrix provides a detailed view of the responsibilities and contributions of each role during the project phase:

Phase	Task	Graphic Designer	Frontend Developer	Backend Developer	QA Tester	DevOps Engineer	Project Manager
Design Phase	Create Wireframes and UI Designs	Primary	Supporting	-	-	-	Approves
	Collect User feedback	Primary	Supporting	-	-	-	Supports
Development Phase	Implement UI functionality	-	Primary	Supporting	-	-	Monitors
	Develop backend logic	-	-	Primary	Supporting	-	Monitors
	Conduct unit testing	-	Supporting	Supporting	Primary	-	Monitors
Testing Phase	Conduct Performance testing	-	-	Supporting	Primary	Supporting	Monitors
Deployment Phase	Manage CI/CD pipeline	-	-	-	-	Primary	Monitors
	Handle system monitoring	-	-	-	Supporting	Primary	Supports

6.Quality Management Plan

Purpose

The QMP establishes the basis for ensuring quality throughout the complete life cycle of the POS System Development project. In line with project management institute standards, the plan emphasizes a commitment to satisfying stakeholder requirements and upholding high software quality. This will enable conformance with performance expectations and usability standardization as elaborated in the project charter.[5]

Quality Objectives

Primary Quality Objectives :The project's quality goals derive from the core requirements specified in the project charter and established industry guidelines:

- **System uptime**: Attaining that 99.9% uptime for POS systems, not a nicety but a precondition for the success of the transaction process and commercial function.

- **Transaction processing time**: Such process time, if kept below two seconds, heightens customer experience during comparison against transaction timing.

- **Real-time synchronization of inventory**: Allow for a maximum lag of 30 seconds for the system to update its inventory in real-time. This plays a crucial role in actuality in the management of stocks.

- **Zero critical defects in production releases**: Zeros critical defects sought-after to ensure system integrity.

- **Code coverage**: Improves the testing and quality of the new features with coverage above 85%.

- **Accessibility compliance**: The interface shall conform to all applicable WCAG 2.1 standards to ensure user inclusiveness and usability .[2]

Quality Metrics

The following table outlines the key quality metrics and their respective targets, along with the measurement methods employed to ensure that the project objectives are consistently met and validated throughout the development lifecycle.[5]

Metric	Target	Measurement Method
System Uptime	99.9%	Automated monitoring tools
Transaction Processing Time	< 2 seconds	Performance testing tools
Code Coverage	> 85%	Unit testing reports
Critical Defects	0 in production	Bug tracking system
User Satisfaction	> 4.5/5	UAT feedback surveys
Database Sync Time	< 30 seconds	System logs analysis

Quality Assurance Activities

Development Phase

The development phase emphasizes preventive measures:

- **Code Reviews**: Peer reviews ensure adherence to coding standards and best practices.
- **Static Code Analysis**: Automation ingeniously performed through GitHub Actions to find potential issues at an early stage.
- **Technical debt appraisal**: Regular assessment prevents the accumulation of dirty code
- **Pair programming**: Since two brains work better than one, development skills are improved; the result is more quality in complex features
- **Checkpoint Integration Daily**: The project is integrated frequently to lessen integration risks

These practices ensure issues are detected and resolved early, promoting efficiency and reliability.

Compliance and Standards

Industry standards.

Key industry standards underpin the Quality Management Plan to ensure compliance and aspects of security.

- ISO 9001:2015: Establishes a stringent framework for quality management.[3]
- OWASP Guidelines: Addresses software security weaknesses.
- PCI DSS: Armor against injury in handling payment transactions.[4]
- GDPR Conformance: Protect user data in accordance with the law of privacy.[2]

Internal Standards

In addition to industry norms, internal standards guide project execution:

- **Coding Standard**s: Promote code domicile with consistency and ease of maintenance.
- **Documentation Standards**: Clear and complete project documentation and reporting.
- **Testing Standards**: State test procedures to verify quality.
- **Security Standards**: Enhance their protection from potential threats to the system.

7.Risk Management

Risk Identification

Risks were identified through:

- Brainstorming with key stakeholders.

- An examination of past data from related projects

- For expert consensus on possible risks use the Delphi Method

Potential risks associated with this project were identified using a combination of approaches, including brainstorming with key stakeholders, examining past data from related projects, and employing the Delphi Method for expert consensus, alongside leveraging the Work Breakdown Structure (WBS) and as a result Risk Breakdown Structure (RBS) is created.

Identified Risks and Descriptions

1. **Incomplete Stakeholder Requirements**

   Description: Stakeholders may fail to articulate clear requirements, leading to missed objectives and scope creep. According to PMI, "unclear or ambiguous requirements lead to missed objectives and often contribute to scope creep" (PMI, 2019).

2. **Technical Design Constraints**

   Description: The project may encounter challenges in implementing the technical architecture due to evolving requirements or unforeseen dependencies between system components. These constraints can lead to inefficiencies in development workflows or bottlenecks in integrating various modules.

3. **Software development delays**

   Description: Unforeseen technical challenges, resource shortages, or inefficient task management may cause delays in the development phase. According to AuditBoard (2024), 50% of software projects face this risk.[9]

4. **Inadequate test coverage**

   Description: Post-deployment problems caused by inadequate testing can raise expenses and have an impact on user satisfaction. This is a major risk for IT projects.

5. **Service disruption in cloud server**

   Description: Outages of cloud services have the potential to disrupt business operations. Dependency on third-party cloud services adds external risk factors according to research by PMI (2023).

6. **Regulatory Changes**

   Description: Unexpected changes in laws or industry standards can affect compliance. This is a common risk across industries.

7. **Insufficient Staffing Levels**

   Description: Resource shortages during critical phases, such as development, can hinder progress. PMI (2023) highlights resource allocation as a common risk in project management highlighting the concerns in software development.

8. **Inaccurate Cost Estimation**

Description: Poor estimation techniques or overlooked costs can result in budget shortfalls. PMI (2023) recommends detailed analysis to mitigate this risk.

9. **Inefficient Communication Channels**

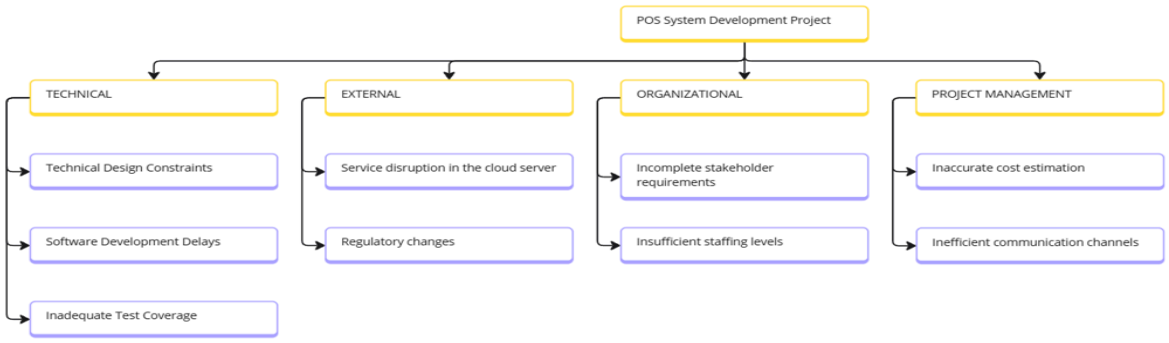
Description: Misaligned communication across teams can lead to delays and misunderstandings, particularly during development. Efficient communication is critical, as noted in PMI best practices (PMI, 2023).

10. **Service disruption in cloud server**

Description: Outages of cloud services have the potential to disrupt business operations. Dependency on third-party cloud services adds external risk factors according to research by PMI (2023).

After identifying the potential risks, a risk-based structure has been created.

Risk Breakdown Structure



Risk Assessment

After identifying the risks, the risks are assessed through an impact table and risk assessment form their impact, likelihood and detection has been identified and their severity has been analyzed. Later a risk severity matrix has been formed with the gathered information.

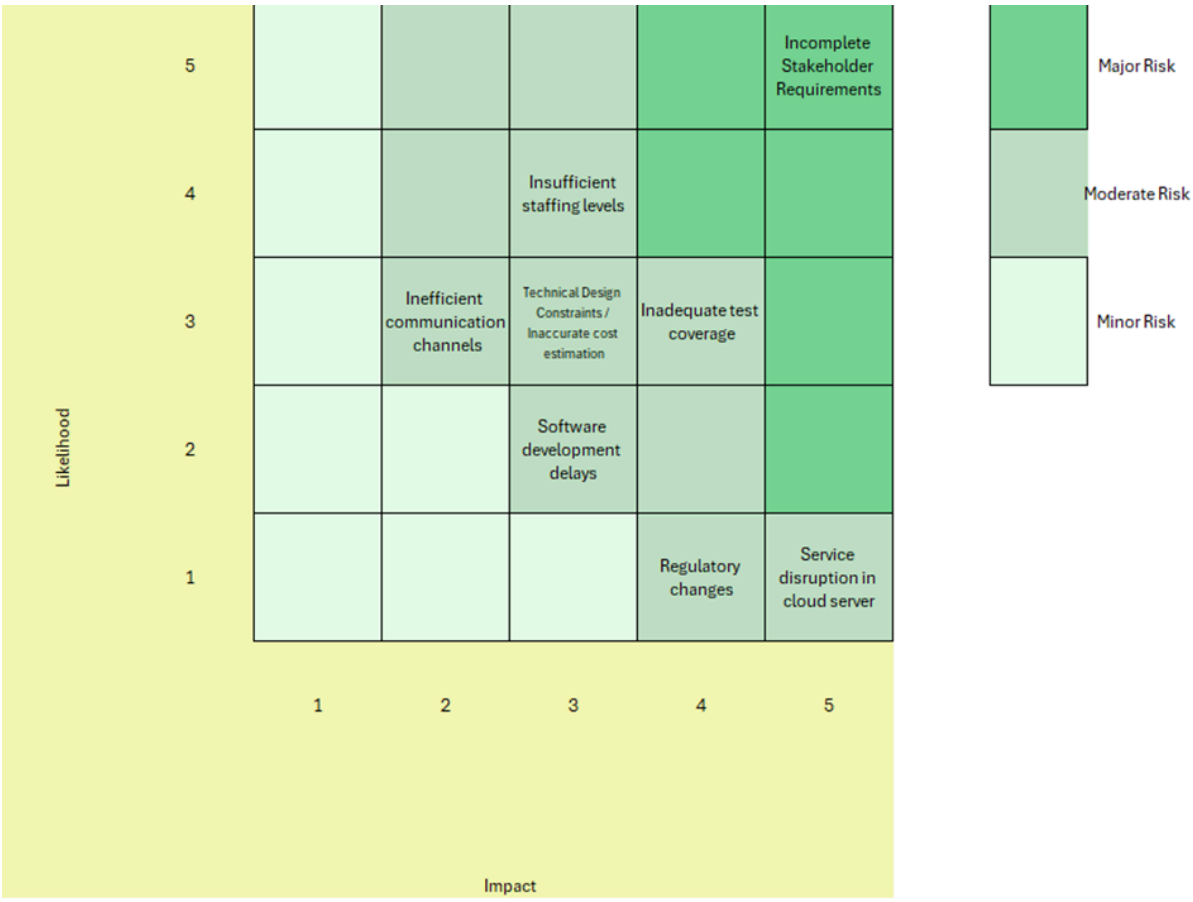
Impact Table

Numeric Scale					
Objective	1 Very Low	2 Low	3 Moderate	4 High	5 Very High
Cost	Insignificant Increase	< 5% Increase	5-10% Increase	10-25% Increase	> 25% Increase
Time	Insignificant Increase	<10% Increase	10-20% Increase	20-50% Increase	> 50% Increase
Scope	Barely Noticable Change	Minor Scope Reduction	Major Scope Reduction	Unapproval Scope Reduction	Scope and Project Total Misalignment

Risk Assessment Form

Risk Event	Likelihood	Impact	Detection Difficulty	Phase
Incomplete stakeholder requirements	5	5	2	Initiation
Technical design constraints	3	3	5	Design
Software development delays	3	3	5	Development
Inadequate test coverage	3	4	5	Testing
Service disruption in cloud server	1	4	1	Deployment
Regulatory changes	1	4	1	Anytime Throughout
Insufficient staffing levels	3	4	3	Development
Inaccurate cost estimation	3	3	3	Initiation
Inefficient communication channels	3	3	3	Anytime Throughout (Mostly Development)

Risk Severity Matrix



Risk Response Development

After the risks were identified and assessed, risk register and risk response matrix were created alongside with risk responses and contingency plans.

Developed Risk Responses

1. **Incomplete Stakeholder Requirements**

   Risk Response: Conduct regular stakeholder meetings, with structured feedback sessions to ensure alignment. Document all requirements in detail and establish sign-off checkpoints.

2. **Technical Design Constraints**

   Risk Response: Focus on modular design principles to isolate and address potential constraints during implementation. Utilize expert reviews and collaborative sessions with design and development teams to identify and resolve constraints proactively, without requiring significant rework.

3. **Software development delays**

   Risk Response: Assign extra team members as needed and include buffer times in the development schedule. Track and prioritize deliverables with Agile sprints.

4. **Inadequate test coverage**

   Risk Response: Use automated testing in addition to human testing. Set aside more time in the calendar for thorough end-to-end and regression testing.

5. **Service disruption in cloud server**

Risk Response: Use SLAs to negotiate with cloud providers for guaranteed uptime. In order to minimize downtime during important disruptions keep an on-premise failover system in place.

6. **Regulatory Changes**

Risk Response: Create a compliance monitoring team and subscribe to regulatory update services. Build flexibility into project processes to accommodate changes as needed.

7. **Insufficient Staffing Levels**

Risk Response: Use temporary staffing agencies to bridge gaps. Implement cross-training programs to enable existing staff to cover additional responsibilities if needed.

8. **Inaccurate Cost Estimation**

Risk Response: Use advanced estimation tools and involve financial experts in the estimation process. Regularly compare actual costs with projections and refine estimates as needed.

9. **Inefficient Communication Channels**

Risk Response: Establish clear communication protocols and provide training on collaboration tools. Schedule regular status meetings to ensure alignment and address any miscommunications promptly.

10. **Service disruption in cloud server**

Risk Response: Use SLAs to negotiate with cloud providers for guaranteed uptime. In order to minimize downtime during important disruptions keep an on-premises failover system in place.

Risk Response Matrix

Risk Event	Response Strategy	Risk Response	Contingency Plan	Who is Responsible
Incomplete Stakeholder Requirements	Avoid	Frequent stakeholder reviews and validations	Reallocate resources to document and address evolving requirements	Project Manager
Technical Design Constraints	Mitigate	Modular design principles and collaborative review sessions.	Engage external experts to resolve unexpected dependencies.	Design Lead
Software development delays	Mitigate	Agile sprints and progress monitoring	Reallocate resources from other phases or hire temporary staff or redistribute tasks among existing staff.	Development lead
Insufficient staffing levels	Mitigate	Temporary staffing and cross-training programs	Reallocate resources from other phases or hire temporary staff or redistribute tasks among existing staff.	HR Manager
Inadequate test coverage	Mitigate	Comprehensive test plans and peer reviews, use of automated tests alongside manual testing	Engage third-party QA services to handle testing gaps.	QA Lead
Service disruption in cloud server	Transfer	SLA agreements with cloud providers	Switch to Backup Cloud Providers	Vendor Manager
Regulatory changes	Escalate	Compliance monitoring team and flexible processes	Allocate resources for immediate compliance adjustments or legal consultations.	Legal Advisor
Inefficient communication channels	Mitigate	Use a comprehensive tool like Slack to centralize communication, with well-defined channels, regular status updates, and real-time collaboration features with predetermined standardized format of messages .	Assign the project manager as a communication facilitator to monitor Slack usage and ensure adherence to protocols.	Project Manager
Inaccurate cost estimation	Mitigate	Use of Function Point Analysis early in the project to obtain an accurate estimation of the project cost	Adjust scope or reallocate funds to cover unforeseen costs.	Finance Lead

Risk Registers

Associated Risk	Category	Response Strategy	Risk Response	Owner	Status
Incomplete Stakeholder Requirements	Organizational	Avoid	Frequent stakeholder reviews and validations	Project Manager	Open
Technical Design Constraints	Technical	Mitigate	Conduct early design validation sessions	Development Team	Open
Software development delays	Technical	Mitigate	Agile sprints and progress monitoring	Development lead	Open
Inadequate test coverage	Technical	Mitigate	Comprehensive test plans and peer reviews	QA Lead	Open
Service disruption in cloud server	External	Transfer	SLA agreements with cloud providers	Project Manager	Open
Regulatory changes	External	Escalate	Regular compliance reviews	Project Manager	Open
Insufficient staffing levels	Organizational	Mitigate	Recruitment and reallocation of resources	Project Manager	Open
Inefficient communication channels	Project Management	Mitigate	Improved tools and training for communication	Project Manager	Open
Inaccurate cost estimation	Project Management	Mitigate	Use of Function Point Analysis early in the project to obtain an accurate estimation of the project cost	Project Manager	Open

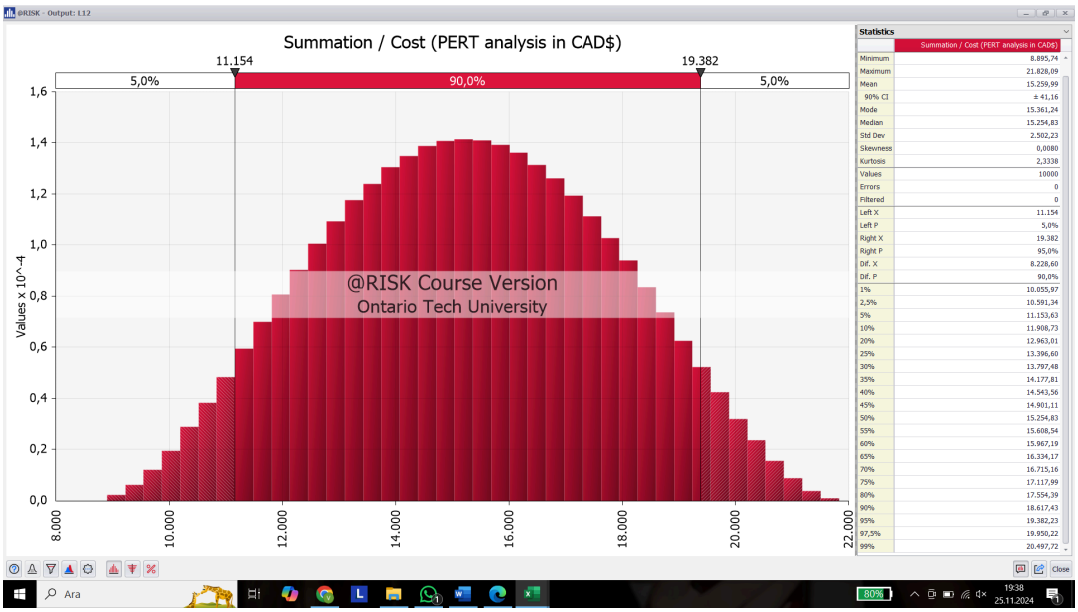
Contingency Reserves

The outcome of risks in case of emerging were calculated. Possible delays were identified and cost estimations were calculated derived from possible delays. Here is the breakdown of possible outcomes of risks in case of emerging prepared in an Excel Spreadsheet.

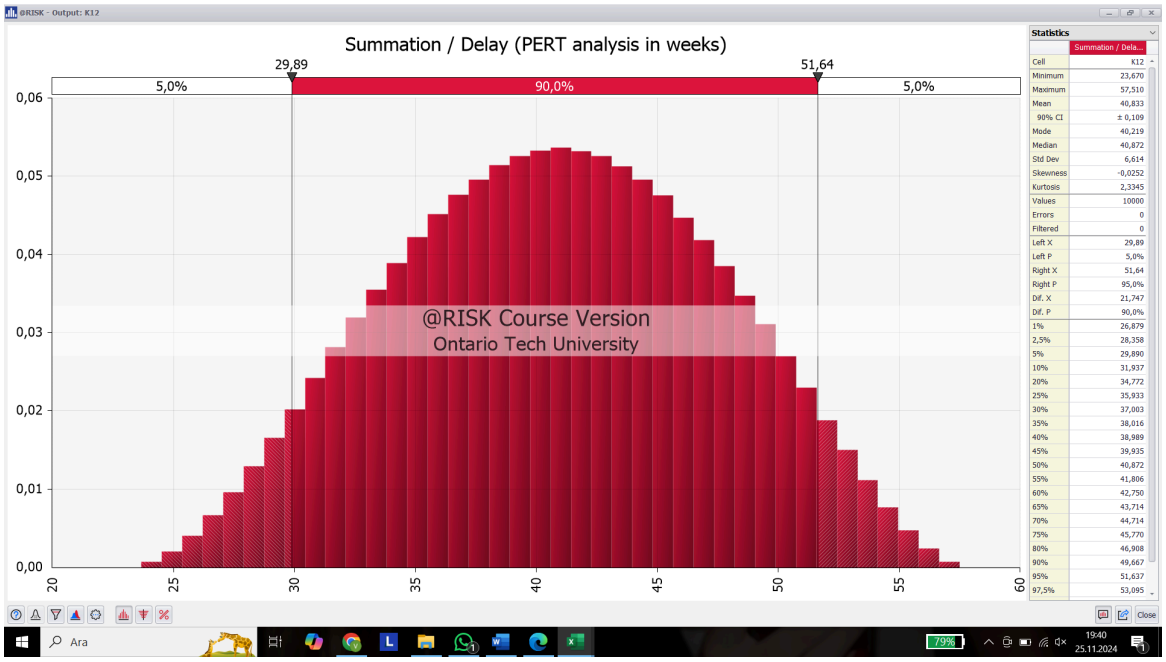
Associated Risk	Involving Team Member	Hourly Rate (CAD$)	Impact	Delay Impact (weeks)			Cost Impact (CAD $)			Delay (PERT analysis in weeks)	Cost (PERT analysis in CAD$)
				Min	Max	Mean	Min	Max	Mean		
Incomplete Stakeholder Requirements	Project Manager	55	Both	4	8	6	1760,00	3520,00	2640,00	6,00	2640,00
Technical Design Constraints	Development lead	40	Cost	4	10	8	1280,00	3200,00	2560,00	7,67	2453,33
Software development delays	Development lead	40	Delay	3	10	7	960,00	3200,00	2240,00	6,83	2186,67
Inadequate test coverage	QA Lead	30	Both	3	6	5	720,00	1440,00	1200,00	4,83	1160,00
Service disruption in cloud server	Deployment Lead	50	Cost	-	-	-	-	-	-	0,00	0,00
Regulatory changes	Project Manager	55	Both	2	8	3	880,00	3520,00	1320,00	3,67	1613,33
Insufficient staffing levels	Project Manager	55	Both	4	8	6	1760,00	3520,00	2640,00	6,00	2640,00
Inaccurate cost estimation	Project Manager	55	Cost	-	-	-	-	-	-	0,00	0,00
Inefficient communication channels	Project Manager	55	Both	3	8	6	1320,00	3520,00	2640,00	5,83	2566,67
Summation				23,00	58,00	41,00	8680,00	21920,00	15240,00	40,83	15260,00

The risks were modeled using PERT analysis. The model was run with 10.000 iterations to produce the most accurate results. @RISK application was used for running the probability model. Below are the results.

Quantitative cost analysis:



Quantitative delay analysis:



Since the software development projects contain a high probability of variability, 90% probability is taken into consideration. From the @RISK cost analysis presented above, 18.617 CAD$ is the amount to cover the 90% probability of the risk model. So the allocated contingency funds decided to be 18.617 CAD$.

Management Reserves

Management reserves will be set at 10% of the total project cost to account for unforeseen risks and major changes that could arise throughout the project. These reserves are distinct from contingency reserves and are specifically allocated for risks associated with the entire project, requiring approval from higher-level management for use. The total project costs were calculated to be $129,183.76. Thus, the allocated amount for management reserves is determined to be $12,918.37.

Total Reserves

Total reserves for foreseen and unforeseen risks are $31,535.37 CAD$.

Risk Monitoring and Control

Techniques:

1. Risk Audits:- Regularly scheduled audits to ensure risk responses are effective.

2. Variance Analysis:- Comparing planned versus actual outcomes to detect emerging risks.

3. Triggers and Thresholds:- Establishing pre-defined indicators for initiating response plans such as KRI's (Key Risk Indicator). Key Risk Indicators are very useful especially in Agile management. Sprint velocities will be compared with pre-defined KRI's to notice emerging risks during the development, testing, and deployment phases.

Tools:

- - Risk Registers: Risk registers will be updated frequently as the project proceeds.
- - Dashboards: Use of Jira dashboards for tracking events with real-time data for detecting riskiness of the event.

Assumptions

1. Risk responses will align with the project’s timeline and budget.

2. Stakeholders will actively participate in risk management activities.

Conclusion

The Risk Management Plan equips the project team with structured tools and strategies to address uncertainties proactively. Regular monitoring ensures risks are managed effectively, safeguarding the project's objectives and stakeholder satisfaction.

8.Communication management Plan

Internal Communication Plan

Purpose

To establish clear guidelines for communication within the internal team, ensuring effective collaboration, efficient task management, and smooth execution of the POS System Development project.

Communication Channels and Tools

TooL	Purpose	Frequency	Responsible Party
Slack	- Real-time updates on project status - Task-specific discussions - Team announcements	Daily	All Team Members
Zoom/Microsoft	- Online team meetings - Weekly progress discussions - Problem-solving sessions	Weekly/As Needed	Project Manager, Team Leads
JIRA	- Tracking sprints, tasks, and issues - Monitoring project progress	Weekly/As Needed	All Team Members

	- Documenting requirements		
GitHub	- Code versioning and CI/CD - Collaboration on software development - Deployment management	Continuous	Development Team, QA Team

Communication Guidelines

1. **Daily Communication**
    - **Tool:** Slack
    - **Purpose:** Share progress updates, report blockers, and collaborate on tasks.
    - **Participants:** All team members
    - **Frequency:** Daily
    - **Expectation:** Provide concise updates via the relevant Slack channels, use tagging to notify specific team members for critical tasks.
2. **Weekly Meetings**
    - **Tool:** Zoom or Microsoft Teams
    - **Purpose:** Review weekly progress, address major challenges, and align on upcoming tasks.
    - **Participants:** Project Manager, Team Leads, Relevant Team Members
    - **Frequency:** Weekly
    - **Expectation:** Prepare a brief update on your team's deliverables before the meeting.
3. **Sprint Planning and Retrospective**
    - **Tool:** JIRA, Zoom or Microsoft Teams
    - **Purpose:** Define sprint tasks, assign responsibilities, and review completed tasks and lessons learned.
    - **Participants:** Project Manager, Development Team, QA Team
    - **Frequency:** Bi-weekly (start and end of each sprint)
    - **Expectation:** All team members update JIRA tasks before sprint reviews.

4. **Code Reviews and Deployment Discussions**
    - **Tool:** GitHub, Slack
    - **Purpose:** Ensure high-quality code and manage deployment pipelines effectively.
    - **Participants:** Development Team, QA Team
    - **Frequency:** Continuous (based on task completion)
    - **Expectation:** Use GitHub pull requests for peer reviews, and Slack for coordination.
5. **Documentation Sharing**
    - **Tool:** JIRA, Confluence, GitHub
    - **Purpose:** Share technical specifications, meeting notes, and development guidelines.
    - **Participants:** All Team Members
    - **Frequency:** Continuous
    - **Expectation:** Upload or update documentation immediately after task completion or meetings.

Collaboration Practices

1. **Slack Channel Structure [1]:**
    - **#general:** General project announcements.
    - **#design-team:** Discussions for the Design Team.
    - **#dev-team:** Development-related conversations.
    - **#qa-team:** Testing updates and bug tracking.
    - **#deployment:** Deployment-related discussions.
    - **#emergency:** For urgent issues requiring immediate attention.
2. **Best Practices for Tools:**
    - **Slack:** Use appropriate channels to avoid information overload.
    - **Zoom/Microsoft Teams:** Always share meeting agendas 24 hours prior.
    - **JIRA:** Update task statuses promptly to ensure accurate sprint tracking.
    - **GitHub:** Provide detailed commit messages and use branch names aligned with tasks in JIRA.

External Communication Plan

Purpose

This communication plan will guide effective engagement with and among all relevant stakeholders involved within the framework of the project. Through addressing specific communication needs, utilizing appropriate media, and maintaining consistent frequency, the current plan will afford timely updates, gather valuable feedback, and create collaboration among concerned parties. Each category of stakeholders would be provided with suitable communications so that they could support their activities, provide continual alignment towards achieving project objectives, and contribute to the successful completion of the initiative.

Stakeholder Communication Details

Stakeholder	Communication Needs	Medium	Frequency	Responsible Party
Warehouse Staff	- Real-time updates on inventory module status - Timeline for delivery of features related to stock management	Emails, Reports	Weekly	Project Manager, QA Tester
Retail Store Managers	- User-friendly interface updates - Progress on data analytics features - Training schedules	Demonstrations, Emails	Bi-weekly	Team Lead: Design Team, UX/UI Designer
IT Department	- Technical documentation - API integration progress - Issue resolution updates	Meetings, Emails	Weekly	Team Lead: Development Team
HQ Management	- Sales analytics and reporting tool progress - Milestone updates - Feedback on strategic alignment	Reports, Presentations	Monthly	Project Manager

Client Companies	Milestones and deliverables updates - Budget and timeline updates - Regular status reports	Reports, Meetings	Monthly	Project Manager
End Customers	- Indirect feedback collected from usability tests - Updates on anticipated improvements to shopping experience	Surveys, Feedback Forms	After User Testing Phases	Team Lead: Testing Team
Store Employees	- Training sessions on POS system - Support for intuitive interface updates - Documentation for daily operational use	Training, User Manuals	Before Deployment, Weekly	Team Lead: Deployment Team

9.Stakeholder management Plan

The key stakeholders in the development of the POS system are as follows

- **Warehouse staff**

  Warehouse workers are responsible for providing inventory cycle counting and movement of materials among warehouses and retail stores. They pull real-time inventory information and synchronization to be time-sensitive so as to prevent stockout and overstocking, making order fulfillment more efficient and accurate.

- **Retail Store Managers**

  Store managers are those who are responsible for a retail outlet's day-to-day operations. For transactions, inventory updates, and analytics for decision-making, a POS system is indispensable to them. They require a system that will give them easy access to necessary data so they can manage the store more efficiently.

- **IT Department**

  The IT department is vital in supervising the integration of the POS system into the general IT infrastructure. They oversee system maintenance, check for compatibility issues, troubleshoot technical problems, and generally make sure that operations continue to be both seamless and secure after deployment.

- **HQ Management**

  Using the reporting and analytic tools of the sophisticated POS system, the headquarters management is able to monitor sales trends and formulate larger strategic decisions. They need a comprehensive, accurate data flow in order to optimize operations across multiple stores and streamline processes, as well as identify growth opportunities.

- **Client Companies**

  Client companies act as the project sponsors, executing requisite funding and offering oversight. They define high-level requirements, approve deliverables, ensure that the POS system is in line with their wider business goals and operational strategies. Their input is critical throughout the project life cycle.

- **End Customers**

  End customers will benefit indirectly from the implementation of the POS system in their businesses. The end users will enjoy the advantages of quick transaction processing, accurate billing, along with a smooth shopping experience, resulting in increased customer satisfaction and loyalty.

- **Store Employees**

  The POS system will be primarily used by the store associates, who operate this system for effective checkout, and this is for the quick and accurate updating of inventory and toiling of the reward and loyalty programs. The interface should be simple so that it becomes an intuitive approach to lessen errors but also ensure smooth running of daily activities.

The table below shows the impact of every stakeholder and their respective roles.

Stakeholder	Impact	Role
Warehouse Staff	High impact: Depend on real-time inventory updates for precise stock management.	Use the system to track inventory levels and coordinate stock movement.
Retail Store Managers	High impact: Need analytics and inventory management for smooth retail operations.	Oversee POS system usage in stores and provide feedback for improvements.
IT Department	Medium impact: Ensure smooth integration and maintain system functionality.	Troubleshoot issues, implement security, and manage system maintenance post-deployment.
HQ Department	High impact: Relies on analytics for decision-making and operational oversight.	Use reports for monitoring sales trends and optimizing business strategies.
Client Companies	High impact: Sponsors and decision-makers; ensure project aligns with business goals.	Provide funding, define requirements, and approve deliverables.
End Customers	Medium impact: Indirect users benefiting from improved service.	Experience the results of a faster, more reliable system during transactions.
Store Employees	High impact: Primary users of	Operate the system to process

	the POS system for daily operations.	transactions, update inventory, and manage customer data.

Stakeholder Engagement Plan

Every identified stakeholder is very critical and the engagement method and level of detail provided to them will vary based on the role they plan and how they impact the development process. The table below shows the engagement plan for the stakeholders.

Stakeholder	Engagement Method	Frequency of Updates	Level of Detail Provided
Warehouse Staff	- Participate in sprint reviews for inventory management functionalities - Provide feedback on inventory modules during UAT	- End of every sprint involving inventory features. - During UAT and pre-deployment phases.	- Detailed insights on real-time stock synchronization workflows and user interface (UI) training.
Retail Store Managers	- Involved in sprint planning for store-related operations and analytics modules. - Usability testing of transaction modules during UAT.	- Bi-weekly updates during sprint retrospectives. - After relevant sprints and during system testing phases.	- Comprehensive details about operational workflows, dashboards, and analytics. - Training materials tailored to store-level operations post-testing.
IT Department	- Included in sprint planning for technical integration and maintenance requirements. - Review system security and compatibility during testing.	-Continuous engagement during development and integration sprints. - End of integration-related sprints and during UAT.	- Full technical documentation covering system architecture, compatibility, and troubleshooting guidelines.
HQ Management	- Participate in sprint reviews for analytics and reporting	- Monthly updates via sprint summaries. - After each major	-High-level summaries of progress, along with

	functionalities. - Provide strategic feedback during milestone reviews.	release or milestone.	tailored demos of analytics dashboards.
Client Companies	- Oversight via sprint demos and milestone reviews. - Final approval of deliverables after release.	- End of every sprint and milestone. - At the completion of each major iteration.	- Executive-level summaries of progress, key decisions, and risks.
End Customers	- Indirect engagement through usability testing involving retail staff.	-During UAT phases.	-No direct engagement; their feedback is inferred through staff input and prototype evaluations.
Store Employees	- Participate in usability testing and sprint reviews for transactional workflows.	- After relevant sprints and during UAT.	- Simplified training materials and user guides for daily operations.

10.REFERENCES

[1] Standuply. (2019). How to use Slack effectively in 2019. In *Standuply*. https://standuply.com/How_to_Use_Slack_Effectively_in_2019_by_Standuply.pdf

[2]Project Management Institute. (2021). A Guide to the Project Management Body of Knowledge (PMBOK® Guide) (7th ed.). Project Management Institute.

[3]International Organization for Standardization. (2015). ISO 9001:2015 Quality Management Systems - Requirements. ISO.

[4]Payment Card Industry Security Standards Council. (2022). Payment Card Industry Data Security Standard (PCI DSS) v4.0.

[5]Ebert, C. & Vector Consulting Services GmbH. (2021). Software Quality Management. In *Vector Consulting Services GmbH*. https://cdn.vector.com/cms/content/consulting/publications/Ebert_QualityManagement.pdf

[6]Schwaber, K., & Sutherland, J. (2020). The Scrum Guide: The definitive guide to Scrum. Scrum.org.

[7]Kerzner, H., & Kerzner, H. R. (2017). Project management: A systems approach to planning, scheduling, and controlling (12th ed.). John Wiley & Sons.

[8]Goldratt, E. M. (2017). Critical chain: A business novel. Routledge.

[9]AuditBoard. (2024). *10 Types of Risk Management Strategies to Follow*. Retrieved from [https://www.auditboard.com/blog/10-risk-management-strategies/] (https://www.auditboard.com/blog/10-risk-management-strategies/)

[10]Glassdoor. (n.d.). *Intermediate software developer salary in Toronto, ON*. Retrieved November 25, 2024, from https://www.glassdoor.ca/Salaries/toronto-on-intermediate-software-developer-salary-SRCH_IL.0,10_IM976_KO11,42.htm

[11]Atlassian. (n.d.). *Jira Software pricing*. Atlassian. Retrieved November 25, 2024, from https://www.atlassian.com/software/jira/pricing

[12]Figma. (n.d.). *Pricing*. Figma. Retrieved November 25, 2024, from https://www.figma.com/pricing/#figma

[13]GitHub. (n.d.). *Pricing*. GitHub. Retrieved November 25, 2024, from https://github.com/pricing

[14]Slack. (n.d.). *Pricing*. Slack. Retrieved November 25, 2024, from https://slack.com/pricing

[15]Zoom. (n.d.). *Plans & pricing*. Zoom. Retrieved November 25, 2024, from https://zoom.us/pricing

[16]Glassdoor. (n.d.). *QA tester salary in Toronto, ON*. Glassdoor. Retrieved November 25, 2024, from https://www.glassdoor.ca/Salaries/toronto-on-qa-tester-salary-SRCH_IL.0,10_IM976_KO11,20.htm

[17]Glassdoor. (n.d.). *UI/UX designer salary in Toronto, ON*. Glassdoor. Retrieved November 25, 2024, from https://www.glassdoor.ca/Salaries/toronto-on-ui-ux-designer-salary-SRCH_IL.0,10_IM976_KO11,25.htm

[18]Glassdoor. (n.d.). *Project manager salary in Toronto, ON*. Glassdoor. Retrieved November 25, 2024, from https://www.glassdoor.ca/Salaries/toronto-on-project-manager-salary-SRCH_IL.0,10_IM976_KO11,26.htm

[19]Glassdoor. (n.d.). *DevOps salary in Toronto, ON*. Glassdoor. Retrieved November 25, 2024, from https://www.glassdoor.ca/Salaries/toronto-on-devops-salary-SRCH_IL.0,10_IM976_KO11,17.htm

[20] **International Function Point Users Group (IFPUG).** *Function Point Counting Practices Manual.* Retrieved from

https://www.ifpug.org/resources/

[21] **Jones, C. (2007).** *Estimating Software Costs: Bringing Realism to Estimating.* McGraw-Hill. ISBN: 978007148300.

[22] **ISO/IEC 20926:2009.** *Software and Systems Engineering* – Function Point Counting Practices Standard. International Organization for Standardization. Retrieved from

https://www.iso.org/standard/40916.html