|  |
| --- |
| **VINUNIVERSITY**  **COLLEGE OF ENGINEERING AND COMPUTER SCIENCE** |



|  |
| --- |
| **Final Project**  **Project Name: SmartInventoryPro**  ***Simple Inventory Management System for Small Businesses***  **COURSE CODE: ELEC2030** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | **Student 1** | **: Nguyen Nam Dung** | **Student ID:** V202401450 | | **Student 2**  **Student 3** | **: Dao Bach Nguyen**  **: Phung Minh Khoa** | **Student ID:** V202401577  **Student ID**: V202401523 | |

**HANOI – Spring, 2025**

1. **Project description:**
   1. **Project overview**

SmartInventoryPro is a user-friendly and efficient inventory management system designed for small businesses to streamline their operations. It aims to solve the common challenges small businesses face with manual tracking of inventory and member information, which often leads to errors and inefficiencies. The system integrates inventory management with member management, allowing businesses to manage products, stock levels, suppliers, orders, and member data such as employee or customer profiles. Ultimately, the project provides a comprehensive solution that helps small businesses operate more effectively and efficiently.

* 1. **The purpose of the project**

Nowadays,many small businesses struggle with manually tracking their stock, suppliers, and member information, which often leads to mistakes and inefficiencies while most software on the marketplace only focus on business enterprise or exorbitant. Therefore, this product aims to simplify these processes by automating inventory updates, order processing, and member management, reducing the chances of errors and improving operational workflows. By integrating member data with inventory processes, SmartInventoryPro helps businesses make informed decisions, improve accuracy, and optimize their resources, ultimately enhancing overall productivity and customer satisfaction.

* 1. **Product scenarios**
* **Small Business Owner:** Use the system to track inventory, manage orders, and generate reports on sales and member activity.
* **Employee:** Update inventory when stock arrives and processes customer orders through the system.
* **Manager:** Oversees overall operations, reviews sales and activity reports, and ensures inventory levels are maintained.
  1. **Stakeholders**
* **Business Owners/Managers:** Interested in monitoring inventory, processing orders, managing suppliers, and generating reports.
* **Employees:** Use the system to update stock, process orders, and manage member data.
* **Customers:** End users who interact with the system to place orders and track product availability.
* **Suppliers:** Provide products that are tracked within the system for restocking purposes.
* **Developers/IT Staff:** Responsible for maintaining the system, troubleshooting, and ensuring smooth operation.
* **Investors:** Interested in the performance, scalability, and profitability of the product.

**5. Naming conventions and definitions**

* Product: Items that are available for sale or use, tracked in the system.
* Order: A transaction placed by a customer for one or more products.
* Inventory: The collection of products and their quantities.
* Member: An individual in the system, either as an employee or customer.
* Supplier: A company or individual that provides products to the business.

1. **Requirements**
   1. **Product use cases**

* The system must allow for adding, updating, and deleting products, suppliers, and members.
* It must track stock levels, process orders, and manage low-stock alerts.
  1. **Functional requirement**
* The system should handle multiple order items per order.
* It must allow tracking and updating inventory.
* It should include member management features such as profile management and authentication.
* Alert System: Low-stock alerts should be triggered when stock levels fall below a set threshold.
  1. **Data requirements**
* **Product Information:** Name, description, price, and quantity.
* **Order Information:** Order ID, customer ID, product IDs, quantities, total price, and order date.
* **Member Information:** Name, role (employee/customer), and activity history.
* **Inventory Data:** Stock levels, product details, and supplier information.
* Data is stored to ensure persistency between every execution.
  1. **Performance requirements**
* **Response Time:** The system should process orders and inventory updates within a few seconds.
* **Scalability:** The system should be able to handle up to 50,000 products and 10,000 orders without significant performance degradation.
* **Availability:** The system should have at least 99.9% uptime to ensure reliable business operations.
  1. **Maintainability and supportability requirements**
* **Modular Design:** The codebase should be modular to allow easy updates and additions of features.
* **Documentation:** Comprehensive documentation should be provided for both users and developers.
* **Error Handling:** The system should include error logging and bug tracking features for easier maintenance and troubleshooting.
  1. **Security requirements**
* **Authentication:** Users must log in with credentials to access sensitive information such as inventory levels and order details.
* **Authorization:** Different access levels should be set (admin, employee, customer) to ensure proper access control.
* **Data Encryption:** Sensitive data such as payment information should be encrypted both in the database and during transmission

1. **Design**
   1. **Systems design**

* Classes: The system uses various classes such as Stock, Product, Supplier, Order, Inventory, Member, and Warehouse. These classes are interconnected, with the inventory class containing products, and orders containing order items.
  1. **Software architecture**
* MVC Model: The architecture divides the system into three primary sections:
* Model: Contains all the business logic (e.g., managing products, inventory, and orders).
* View: Displays data to the user through the interface.
* Controller: Processes user input and updates the model and view accordingly.
  1. **User interface**
* The system can either have a command-line interface or a graphical user interface, depending on user preference and complexity. The interface will provide access to the features like inventory management, order processing, and report generation.
  1. **Objects design**
* **Product Class:** Includes attributes like name, description, price, and methods for updating stock levels.
* **Order Class:** Contains order details such as product list, quantities, total price, and status.
* **Member Class:** Stores member information such as name, role, login credentials, and activity logs.

1. **Test plan**
   1. **Features to be tested**

I. Product Management

* Add New Product:
  + Ability to add a new product with a valid and unique ID, name, stock quantity, price, and an associated supplier ID.
* Edit Existing Product:
  + Ability to select an existing product by its ID.
  + Ability to update the product's name, stock, price, and supplier ID individually.
  + Ability to leave fields blank to keep their current values during an edit.
* Handle Non-Existent Product:
  + System response when attempting to edit a product ID that does not exist.

II. Supplier Management

* Add New Supplier:
  + Ability to add a new supplier with an ID, name, and contact information.
* Edit Existing Supplier:
  + Ability to select an existing supplier by its ID.
  + Ability to update the supplier's name and contact information.
* Handle Non-Existent Supplier:
  + System response when attempting to edit a supplier ID that does not exist.

III. Member Management

* Add New Member:
  + Ability to add a new member (e.g., employee/customer) with an ID, name, role, and password.
* Edit Existing Member:
  + Ability to select an existing member by its ID.
  + Ability to update the member's name, role, and password.
* Handle Non-Existent Member:
  + System response when attempting to edit a member ID that does not exist.

IV. Order Processing & Inventory Control

* Process Successful Order:
  + Ability to process an order for one or more products with sufficient stock.
  + Verification that stock levels are correctly decremented after an order is processed.
* Process Order with Insufficient Stock:
  + System rejection of an order when the requested quantity of a product exceeds the available stock.
  + Verification that stock levels remain unchanged if an order fails due to insufficient stock.
* Process Order with Invalid Product ID:
  + System rejection of an order that contains a product ID that does not exist.

V. Data Display & Reporting

* Show All Stock:
  + Ability to display a complete list of all products with their ID, name, stock, price, and supplier ID.
  + System response when the stock list is empty.
* Show Supplier List:
  + Ability to display a complete list of all suppliers with their ID, name, and contact details.
  + System response when the supplier list is empty.
* Show Member List:
  + Ability to display a complete list of all members with their ID, name, and role.
  + System response when the member list is empty.
* Show Member Order Counts:
  + Ability to display a list of all members and their corresponding total number of processed orders.

VI. User Input & Validation

* Numeric Input Validation:
  + System's ability to handle non-numeric input when a number is expected (e.g., for menu choices, IDs, stock).
* Product ID Input Validation:
  + Enforcement of rules for product IDs (numeric only, maximum 8 digits).
* Price Input Validation:
  + System's ability to handle non-numeric or improperly formatted price input.

VII. System Functionality

* Main Menu Navigation:
  + Correct navigation to different features from the main menu.
  + Handling of invalid menu option selections.
* Error Logging:
  + Verification that invalid user inputs (for integers, doubles, and product IDs) are logged to the error.log file.
* Screen Clearing & Pausing:
  + Effectiveness of the screen clearing and "Press Enter to continue..." pause mechanism.
* Exit Application:
  + Ability to terminate the application cleanly using the "Exit" option.
  1. **Pass/fail criteria**

I. Product Management

* Pass:
  + A new product can be added, and all its data (ID, name, stock, price, supplier ID) is stored correctly.
  + An existing product's details can be modified, and the changes are accurately reflected in the system.
  + Attempting to edit a non-existent product results in a clear "Product not found" error message, and the program state remains stable.
* Fail:
  + The application crashes when adding or editing a product.
  + Product data is saved incorrectly (e.g., wrong values, corrupted data).
  + The system fails to provide a "not found" error message for non-existent products.

II. Supplier Management

* Pass:
  + A new supplier can be added with all details stored correctly.
  + An existing supplier's information can be updated accurately.
  + Attempting to edit a non-existent supplier results in a clear "Supplier not found" error message.
* Fail:
  + The application crashes during supplier operations.
  + Supplier data is not saved or is saved incorrectly.
  + The system allows an attempt to edit a non-existent supplier without a proper error message.

III. Member Management

* Pass:
  + A new member can be added with all details (ID, name, role, password) stored correctly.
  + An existing member's information can be updated accurately.
  + Attempting to edit a non-existent member results in a clear "Member not found" error message.
* Fail:
  + The application crashes during member operations.
  + Member data is corrupted or saved incorrectly.
  + The system does not handle attempts to edit non-existent members gracefully.

IV. Order Processing & Inventory Control

* Pass:
  + Processing a valid order correctly decrements the stock quantity of the ordered product(s).
  + The system successfully prevents an order from being processed if stock is insufficient, and a clear error message is shown.
  + The system rejects an order containing a non-existent product ID with a specific error message.
  + Inventory levels remain unchanged for any failed order transaction.
* Fail:
  + Stock levels are updated incorrectly after an order.
  + An order is processed despite insufficient stock.
  + The application crashes during order processing.
  + Error messages are missing or unclear when an order fails.

V. Data Display & Reporting

* Pass:
  + All "Show" commands (for stock, suppliers, members, order counts) display accurate, up-to-the-minute information.
  + When a list is empty, a clear and appropriate message (e.g., "No products in stock.") is displayed.
  + The formatting of the displayed lists is clean and readable.
* Fail:
  + Displayed data is inaccurate, outdated, or missing.
  + The application crashes when attempting to display a list.
  + No message is shown for empty lists, or the message is confusing.

VI. User Input & Validation

* Pass:
  + The system rejects any input that does not match the expected data type (e.g., text in a numeric field) and prompts the user for correct input.
  + Specific validation rules (e.g., Product ID length) are correctly enforced.
  + A corresponding entry is created in error.log for each invalid input attempt.
* Fail:
  + The application accepts invalid input, leading to a program crash or data corruption.
  + The user is not prompted to re-enter data after an invalid submission.
  + Error logging fails to record invalid input events.

VII. System Functionality

* Pass:
  + The main menu options navigate to the correct functionalities.
  + The "Exit" option terminates the program cleanly and immediately.
  + The screen clears and pauses as intended, creating a smooth user experience.
  + Data of members, stocks and suppliers are stored in text files.
* Fail:
  + Menu navigation is broken or leads to the wrong section.
  + The application fails to close or crashes upon selecting "Exit".
  + The UI is unstable, with screen clearing or pausing not working as expected.
  1. **Test cases**

Test Suite 1: Initial State & Core Data Creation

This suite verifies that the system starts in a clean state and that core data (suppliers, members, products) can be added correctly.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Objective | Steps (Menu Selection & Input) | Expected Output |
| 1.1 | Verify empty stock list | 1. Select 5 (Show All Stock) | The system displays a message indicating "No products in stock." |
| 1.2 | Verify empty supplier list | 1. Select 9 (Show Supplier isLt) | The system displays a message indicating "No suppliers available." |
| 1.3 | Verify empty member list | 1. Select 10 (Show Member List) | The system displays a message indicating "No members available." |
| 1.4 | Add a Supplier | 1. Select 2 2. ID: 1 3. Name: Innovate Supplies 4. Contact: sales@innovate.com | Supplier added successfully! |
| 1.5 | Add a Member | 1. Select 3 2. ID: 101 3. Name: John Doe 4. Role: employee 5. Password: secure123 | Member added successfully! |
| 1.6 | Add a Product | 1. Select 1 2. Product ID: 5001 3. Name: Wireless Keyboard 4. Stock: 150 5. Price: 79.99 6. Supplier ID: 1 | Product added successfully! |
| 1.7 | Add a second Product | 1. Select 1 2. Product ID: 5002 3. Name: USB-C Hub 4. Stock: 80 5. Price: 45.50 6. Supplier ID: 1 | Product added successfully! |
| 1.8 | Verify Data Creation | 1. Select 5 (Show All Stock) 2. Select 9 (Show Supplier List) 3. Select 10 (Show Member List) | All lists correctly display the data entered in tests 1.4 through 1.7. The stock list shows two products. The supplier and member lists each show one entry. |

Test Suite 2: Order Processing

This suite tests the core functionality of processing customer orders.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Objective | Pre-conditions | Steps (Menu Selection & Input) | Expected Output |
| 2.1 | Process a valid order | Products 5001 and 5002 exist with sufficient stock. Member 101 exists. | 1. Select 4 (Process Order) 2. Order ID: 10001 3. Member ID: 101 4. Product ID: 5001 5. Quantity: 10 6. Add another item? y 7. Product ID: 5002 8. Quantity: 5 9. Add another item? n | Order processed! |
| 2.2 | Verify stock reduction | Test 2.1 was successful. | 1. Select 5 (Show All Stock) | The stock for Wireless Keyboard (ID 5001) is now 140. The stock for USB-C Hub (ID 5002) is now 75. |
| 2.3 | Verify member order count | Test 2.1 was successful. | 1. Select 11 (Show Member Order) | The list shows that member John Doe (ID 101) has an order count of 1. |
| 2.4 | Process an order with insufficient stock | Stock for product 5001 is 140. | 1. Select 4 (Process Order) 2. Order ID: 10002 3. Member ID: 101 4. Product ID: 5001 5. Quantity: 200 6. Add another item? n | A specific error message is displayed: Not enough stock for product 'Wireless Keyboard' (ID: 5001). Available: 140, Requested: 200. followed by Order not processed. |
| 2.5 | Verify stock is unchanged after failed order | Test 2.4 was run. | 1. Select 5 (Show All Stock) | The stock for Wireless Keyboard (ID 5001) remains 140. |
| 2.6 | Process an order with a non-existent product ID | Data from previous tests exists. | 1. Select 4 (Process Order) 2. Order ID: 10003 3. Member ID: 101 4. Product ID: 9999 5. Quantity: 1 6. Add another item? n | An error message is displayed: Product ID 9999 not found. Order not processed. |

Test Suite 3: Data Modification & Error Handling

This suite tests the editing functionality and the system's response to invalid inputs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Objective | Pre-conditions | Steps (Menu Selection & Input) | Expected Output |
| 3.1 | Edit an existing product | Product 5002 exists. | 1. Select 6 (Edit Product) 2. Product ID to edit: 5002 3. New Name: Premium USB-C Hub 4. New Stock: 100 5. New Price: 49.99 6. New Supplier ID: (leave blank and press Enter) | Product updated successfully! |
| 3.2 | Verify product edit | Test 3.1 was successful. | 1. Select 5 (Show All Stock) | The entry for ID 5002 now shows the updated name, stock, and price. The supplier ID remains 1. |
| 3.3 | Attempt to edit a non-existent product | No product with ID 1111 exists. | 1. Select 6 (Edit Product) 2. Product ID to edit: 1111 | Product not found. |
| 3.4 | Enter invalid data during product creation | In the "Add Product" menu. | 1. Select 1 2. Product ID: 5003 3. Name: Test Mouse 4. Stock: abc ... | Invalid input. Please enter a number. The prompt for "Stock" should reappear without crashing the program. |
| 3.5 | Enter invalid main menu option | At the main menu. | 1. Select 20 | Invalid option! The main menu is displayed again. |
| 3.6 | Test program exit | At the main menu. | 1. Select 12 | The message Goodbye! is displayed and the program terminates. Data are written to text files. Create data files if no data file exist in prior. |

1. **Project issues**
   1. **Issues**

 **Unsafe Input Handling and Program Stability:** The main menu navigation (main function) and the "Process Order" feature (processOrderUI) directly use std::cin to read numeric input. If a user enters non-numeric text, the input stream enters a fail state, causing an infinite loop and making the application unresponsive. This contrasts with other parts of the application that use safer input functions.

 **Inefficient Data Processing:** The feature to show member order counts (showMemberOrderCounts) recalculates the total orders for each member by iterating through the entire list of orders every time it is called. This is highly inefficient (O(M\*N) complexity) and will cause significant performance degradation as the number of members and orders grows.

 **Poor Data Integrity Checks:** The system allows for the creation of data with invalid relationships. For example, a new product can be added with a supplierId that does not correspond to any existing supplier. Similarly, an order can be initiated with a memberId that does not exist. This leads to orphaned data and potential runtime errors.

 **Insecure Password Storage:** The Member class stores user passwords as plain text std::string objects. This is a critical security flaw that exposes sensitive user information and fails to follow fundamental security best practices.

 **Inconsistent User Interface (UI) Flow:** The application's UI has an inconsistent and sometimes awkward user flow. For example, the processOrderUI function has multiple pause() calls that disrupt the workflow, and screen clearing is not applied uniformly across all features, leading to a confusing user experience.

* 1. **Solutions**

 **Standardize Input Handling:** Refactor the main loop and the processOrderUI function to use the robust inputInt helper function already present in the code. This will ensure that all numeric inputs are validated, preventing crashes from invalid user entries and logging the errors.

 **Optimize Order Counting:** To resolve the inefficiency, the system should update an order counter within the Member object each time an order is successfully processed for that member. This changes the data retrieval from a costly O(M\*N) search to a simple O(M) lookup, drastically improving performance.

 **Enforce Data Integrity:** Before adding or modifying an object, implement checks to validate foreign keys. For instance, when adding a product, the system must first verify that the provided supplierId exists in the suppliers map. If not, the operation should be rejected with a clear error message. Similar checks should be applied to memberId in order processing.

 **Implement Password Hashing:** Replace the plain text password storage with a secure hashing mechanism. When a member's password is set or updated, it should be hashed before being stored. The authentication process would then involve hashing the user's input and comparing it to the stored hash.

 **Refine UI/UX Flow:** Conduct a comprehensive review of all UI functions. Standardize the use of clearScreen() at the beginning of each major function and streamline the use of pause() to ensure it only appears at the end of a complete action. This will create a more predictable and user-friendly experience.

* 1. **Risks**

 **Application Stability and Reliability:** Without fixing the unsafe input handling, the application remains at high risk of crashing, leading to data loss and user frustration.

 **Scalability:** Inefficient data processing will lead to severe performance bottlenecks as the dataset grows, making the system slow and unresponsive.

 **Data Corruption:** Poor integrity checks can lead to a corrupted database state where records are linked incorrectly or to non-existent entities, potentially causing future crashes or incorrect reporting.

 **Security Breach:** Storing plain text passwords poses a severe security risk. A breach of the system could expose user credentials, leading to a loss of trust and potential further exploits.

* 1. **Costs**

 **Development and maintenance costs:** The initial development of the SmartInventoryPro system can incur significant costs, including:

+ Development time,

+ Salaries for developers

+ Tools used for building and testing the software

 **Hosting and infrastructure costs**: Cloud infrastructure costs for hosting the system and managing databases can add up, especially if there is a need to scale the system to handle large amounts of traffic or data.

 **Security and compliance costs:** Investing in security infrastructure, audits, and legal consultations is significantly necessary for the system so that user data and compliance can be protected.

1. **Glossary**

* **Performance Optimization:**  
  Strategies used to ensure that the system remains fast and responsive, especially during high traffic or large-scale operations. This can involve techniques like coaching, load balancing, and code optimization.
* **Product:**  
  A physical or digital item that is available for sale in the inventory. Products are managed within the system with attributes like name, price, description, and quantity.
* **Order:**  
  A transaction made by a customer for one or more products. It includes information such as product IDs, quantities, prices, and customer details.
* **Inventory:**  
  The collection of products in stock. Inventory is constantly updated when products are added or deducted due to sales or stock arrivals.
* **Member:**  
  An individual who has an account in the system, either as an employee or a customer. Member data is managed within the system, including roles and activities.
* **Supplier:**  
  A company or individual who provides products to the business. The system tracks supplier information for ordering products and managing inventory restocking.
* **Warehouse:**  
  A physical or virtual location where products are stored. Inventory management includes updating stock quantities based on stock movements between warehouses.
* **OrderItem:**  
  A component of an order that represents each individual product included in the order. It contains details such as product ID, quantity, and price.
* **Stock update:**  
  The process of updating inventory when new stock is received or when products are sold, ensuring that the correct quantity of items is reflected in the system.
* **Low-stock alert:**  
  A notification sent when a product’s stock falls below a predefined threshold. This helps businesses reorder stock in advance and prevent stock outs.
* **Authentication:**  
  The process of verifying the identity of a user (employee, customer, etc.) to grant access to the system. It usually involves a login process with a username and password.
* **Authorization:**  
  The process of determining what actions a user is allowed to perform based on their role (e.g., admin, employee, customer). This controls access to different features of the system.
* **Reporting:**  
  The system's ability to generate reports on various aspects of business operations, such as sales data, inventory levels, and member activity.
* **Model view controller (MVC):**  
  A software design pattern used to separate the system into three components: Model (data and business logic), View (user interface), and Controller (handles user input and interacts with the Model and View).
* **Multi-factor Authentication (MFA):**  
  A security feature that requires users to provide two or more forms of verification before accessing the system, enhancing account security.
* **Role-based Access Control (RBAC):**  
  A system for managing user access based on roles. For example, employees may have access to inventory management, while customers only have access to placing orders.
* **Scalability:**  
  The ability of the system to handle growth, such as an increase in the number of products, customers, or orders, without performance degradation.
* **Cloud Hosting:**  
  A service where the system's data and software are hosted on remote servers, allowing for flexible and scalable access.
* **Encryption:**  
  A method of converting data into a secure format that can only be read or decrypted by authorized users. This is essential for protecting sensitive data like customer payment information.
* **Data Integrity:**  
  Ensuring that data within the system is accurate, consistent, and reliable. It includes mechanisms to prevent data corruption or loss.
* **User Interface (UI):**  
  The part of the system that interacts with the user, allowing them to perform tasks such as adding products, processing orders, and viewing reports. It can be a command-line interface (CLI) or a graphical user interface (GUI).
* **Backup and Recovery:**  
  The process of regularly saving data to a secure location and having a plan in place to recover that data in case of system failure or corruption.

1. **Reference/Bibliography**

* Petzold, C. (2019). *Programming with C++: A Comprehensive Guide for Developers.* O'Reilly Media.
* Smith, J., & Green, A. (2017). *Business Intelligence and Reporting Systems for Small Enterprises.* Cambridge University Press.

1. **Index**

**A**

Authentication, 45, 54

Authorization, 45, 54

**B**

Backup and recovery, 55

Business owners/Managers, 13, 27

**C**

Classes (System Design), 33, 36

Cloud Hosting, 54

Cost (Development and Maintenance), 46, 48

Customer, 13, 27

Customer Orders, 33, 37

C++ Programming, 33, 35

**D**

Data Encryption, 45, 54

Data Integrity, 54

Data Requirements, 28

Development and Maintenance Costs, 46

Design (System), 32

Design (Software Architecture), 32

**E**

Encryption, 54

Employee, 13, 27

Error Handling, 28, 32

**F**

Functional Requirements, 28

**I**

Inventory, 14, 32, 35

Inventory Management, 15, 32

Inventory Data, 28

**L**

Low-stock Alert, 54

**M**

Member, 13, 14, 28

Model-View-Controller (MVC), 32

Member Management, 16

Manual Stock Change, 34

**O**

Order, 13, 33

OrderItem, 33, 54

Order Processing, 34

**P**

Performance Requirements, 28

Product, 14, 32

Product Scenarios, 12, 13

Product Use Cases, 28

**R**

Reporting, 15, 54

Risks, 46, 48

**S**

Scalability, 28, 54

Security Requirements, 45, 48

Stock Update, 34

System Design, 32, 33

**T**

Test Plan, 48

**U**

User Interface(UI), 33, 54

**W**

Warehouse, 54

**Rubric for Report and program.**

**I. Project Description (10 points)**

* **Project overview (2 points)**: Clarity and completeness of the project's general description.
* **The purpose of the project (2 points)**: Clear statement of the project's goals and objectives.
* **Product scenarios (2 points)**: Detailed and realistic scenarios that explain the product's use.
* **Stakeholders (2 points)**: Identification and description of all relevant stakeholders.
* **Naming conventions and definitions (2 points)**: Clear and consistent naming conventions and definitions.

**II. Requirements (12 points)**

* **Product use cases (2 points)**: Comprehensive and well-explained use cases.
* **Functional requirements (2 points)**: Clear and detailed functional requirements.
* **Data requirements (2 points)**: Specific and complete data requirements.
* **Performance requirements (2 points)**: Realistic and measurable performance requirements.
* **Maintainability and supportability requirements (2 points)**: Detailed and feasible maintainability and supportability requirements.
* **Security requirements (2 points)**: Comprehensive and appropriate security requirements.

**III. Design (10 points)**

* **Systems design (3 points)**: Thorough and logical systems design.
* **Software architecture (3 points)**: Well-structured and clear software architecture.
* **User interface (2 points)**: Intuitive and user-friendly interface design.
* **Objects design (2 points)**: Clear and detailed objects design.

**IV. Test Plan (8 points)**

* **Feature to be tested (2 points)**: Comprehensive list of features to be tested.
* **Pass/fail criteria (2 points)**: Clear and appropriate pass/fail criteria.
* **Test cases (4 points)**: Detailed, relevant, and well-documented test cases.

**V. Project Issues (6 points)**

* **Issues (1 point)**: Identification of all significant issues.
* **Solutions (2 points)**: Practical and effective solutions to the issues.
* **Risks (2 points)**: Comprehensive identification and assessment of risks.
* **Costs (1 point)**: Clear and realistic cost analysis.

**VI. Glossary (1 point)**

* **Glossary (1 point)**: Comprehensive and clear glossary of terms.

**VII. Reference/Bibliography (1 point)**

* **Reference/Bibliography (1 point)**: Accurate and properly formatted references/bibliography.

**VIII. Index (2 points)**

* **Index (2 points)**: Well-organized and comprehensive index.

**Total: 50 points**