

American International University-Bangladesh (AIUB)  
**Department of Computer Science  
Faculty of Science & Technology (FST)**

**Blood Donation Management System**

A Software Engineering Project Submitted

By

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Semester: Summer 24-25** | | **Section:** | **Group Number:** | |
| **SL** | **Student Name** | **Student ID** | **Contribution (CO3+CO4)** | Individual Marks |
| 1 |  |  |  |  |
| 2 | MAOUN BILLAH | 22-48813-3 |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |

The project will be evaluated for the following Course Outcomes



|  |  |  |
| --- | --- | --- |
| ***CO3 (PO-g-1)***  ***Select appropriate software engineering models, project management roles and their associated skills for the complex software engineering project and evaluate the sustainability of developed software, taking into consideration the societal and environmental aspects*** | Total Marks | |
|  | |
| Selection of Software Engineering Models: Process model selection and presents sufficient evidence to support argument for the model selection | [5 Marks] |  |
| Role identification and Responsibility Allocation: Well-planned project with proper role identification and responsibility allocation in the project management activities | [5Marks] |  |
| Formatting and Submission: Submission, Defense, Completeness, Spelling, grammar, and Organization of the Project report | [5Marks] |  |
| ***CO4 (PO-k-1)***  ***Apply engineering management principles and economic decision making to develop software engineering project management plan.*** | Total Marks | |
|  | |
| Project WBS and Testcases: Relevant WBS (project task list) and testcases for the proposed project are stated properly. | [5Marks] |  |
| Effort Estimation and Scheduling: Project estimation was described using proper effort estimation or schedules based on available project resources | [5Marks] |  |
| Risk Management: Sufficient and appropriate risks are identified, analyzed, and properly categorized or prioritized. | [5Marks] |  |



# PROJECT PROPOSAL

## Background to the Problem

* Write the background description that helps to show your project into the right context of a problem domain and gives everyone involved a common view of the project.

Blood Bank bd is a platform designed to streamline the entire blood donation lifecycle, connecting all key stakeholders to ensure a more efficient and responsive blood supply. Blood Bank Bd significantly boost donor engagement, minimize blood wastage, enhance emergency response capabilities, and ultimately save more lives by creating a smarter, more interconnected blood donation system.

## Solution to the Problem and Process Model Selection

* Describe what your project scopes and features are.

Users and Features

* Create a user story table and add the user story table here.
* Add the User Story Board that you made using Trello.
* What are the existing software solutions that are available to solve mentioned problem?
* Select one software development process model from the models taught in the course that best suits your project.
* Provide an analysis of the project environment, including the nature of requirements and whether they are stable or likely to change.
* Explain how the selected model supports your team size, communication and coordination of tasks. Is the solution feasible to meet the business objective?
* Evaluate how flexible the model is in adapting to changes in scope, technology, or user requirements.
* Provide deep insight that demonstrate and preset a creative solution to the real‐life problem.
* Describe the target group of users of your solution? And how will they benefit from your proposed solution to the problem?
* Describe the contribution of your project to the development of scientific results that are identified and well documented.
* Presents enough evidence to support argument for your model selection in developing your proposed solution.
* Discuss how the process model manages project risks and uncertainties at different stages.
* Relate the process model to the project schedule, showing how it supports timely delivery and meeting deadlines.
* Lastly, present a justification that explains clearly why the chosen model is more suitable than other alternatives.
* Create a user story table and add the user story table here.

[https://www.notion.so/Project-Requirement-Document-PRD- 248c4085a9578042a52cf9fdea218ae4?source=copy\_link](https://www.notion.so/Project-Requirement-Document-PRD-%20%20248c4085a9578042a52cf9fdea218ae4?source=copy_link)

* Add the User Story Board that you made using Trello.

<https://trello.com/invite/b/688adde9ff1ea6a061a243fb/ATTI53b0454f1985e12ebc3616a5ed86d97738CCF963/bloodlink>

* What are the existing software solutions that are available to solve mentioned problem?
* Select one software development process model from the models taught in the course that best suits your project.

For the Blood Donation Management System, the **Extreme Programming (XP) model** is selected because it supports iterative development with frequent releases, allowing continuous feedback from donors, hospitals, and blood bank staff. XP emphasizes simplicity, flexibility, and adaptability, which is crucial as user needs in healthcare may change. Its focus on continuous testing and integration ensures a reliable, error-free system, while close collaboration with stakeholders keeps the system user-friendly and efficient. Overall, XP provides fast delivery, high quality, and adaptability, making it the ideal choice for this project.

* Provide an analysis of the project environment, including the nature of requirements and whether they are stable or likely to change.

The Blood Donation Management System operates in a dynamic healthcare environment with multiple stakeholders, including donors, hospitals, and blood banks. Core requirements like donor registration and blood inventory are stable, while features such as notifications or analytics may change based on feedback and policy updates. Non-functional needs include reliability, security, and usability. Overall, the environment is moderately dynamic, making an iterative approach like **XP** ideal for adapting to evolving requirements.

* Explain how the selected model supports your team size, communication and coordination of tasks. Is the solution feasible to meet the business objective?

The **XP model** suits our small-to-medium team by promoting close collaboration, daily communication, and coordinated task management. Its iterative development and continuous testing allow the team to adapt quickly to changes. This ensures the system is reliable, user-friendly, and meets the business objective of efficiently managing blood donations.

* Evaluate how flexible the model is in adapting to changes in scope, technology, or user requirements.

The **Extreme Programming (XP) model** is highly flexible in adapting to changes. Its iterative approach allows frequent releases and continuous feedback, making it easy to accommodate evolving user requirements, changes in project scope, or new technologies. Continuous integration and testing ensure that updates can be implemented safely without disrupting existing functionality, making XP ideal for dynamic environments like a Blood Donation Management System.

* Provide deep insight that demonstrate and preset a creative solution to the real‐life problem.

The Blood Donation Management System creatively solves the real-life problem of blood shortages by connecting donors, hospitals, and blood banks in real time. It uses smart matching algorithms based on location, blood type, and urgency, while predictive inventory management forecasts shortages and optimizes stock. Integrated notifications keep donors and staff informed, and data-driven analytics help administrators plan drives and allocate resources efficiently. This combination of technology, real-time communication, and user-friendly design ensures faster blood availability and ultimately saves lives.

* Describe the target group of users of your solution? And how will they benefit from your proposed solution to the problem?

The Blood Donation Management System targets blood donors, hospitals, blood banks, and emergency services. Donors benefit from easy registration, timely notifications, and simplified scheduling. Hospitals and blood banks gain real-time inventory tracking, faster fulfillment of requests, and predictive insights for stock management. Patients and emergency services receive quicker access to needed blood, improving critical outcomes. Overall, the system streamlines communication, enhances efficiency, and helps save lives.

* Describe the contribution of your project to the development of scientific results that are identified and well documented.

The Blood Donation Management System contributes to scientific development by providing a well-documented, data-driven platform for analyzing blood donation patterns, donor behavior, and hospital demand trends. This data supports research in public health and resource optimization, helping identify shortages and regional trends. By systematically recording processes and outcomes, the project creates a reliable dataset for future studies and policy-making, demonstrating how technology can optimize healthcare logistics and improve life-saving operations.

* Presents enough evidence to support argument for your model selection in developing your proposed solution.

The **XP model** is ideal for the Blood Donation Management System because it allows iterative development with frequent releases, adapts easily to changing healthcare requirements, and ensures continuous testing for reliability. Its emphasis on stakeholder collaboration keeps the system aligned with user needs. Evidence from similar healthcare systems shows that iterative, feedback-driven approaches improve quality, reduce errors, and enhance user satisfaction, making XP the best choice for this project.

* Discuss how the process model manages project risks and uncertainties at different stages.

The **XP model** manages project risks through iterative development, continuous testing, and frequent stakeholder feedback. Early clarification of requirements, short development cycles, and automated testing reduce technical, schedule, and quality risks. Frequent releases and collaboration ensure the system adapts to changing needs and meets user expectations, keeping the project reliable and on track.

* Relate the process model to the project schedule, showing how it supports timely delivery and meeting deadlines.

The Extreme Programming (XP) model supports timely delivery through its iterative, shortdevelopment cycles (sprints). Each iteration produces a functional increment, allowing the team to complete tasks in manageable segments and quickly address issues. Continuous integration and frequent testing reduce delays caused by errors. Regular stakeholder feedback ensures priorities are clear, helping the team focus on high-value features first. Overall, XP’s structure promotes on-time delivery and ensures deadlines are met without compromising quality.

* Lastly, present a justification that explains clearly why the chosen model is more suitable than other alternatives.

The Extreme Programming (XP) model is more suitable for the Blood Donation Management System than other alternatives because it combines flexibility, iterativedevelopment, and continuous testing, which are essential for a dynamic healthcare environment. Unlike Waterfall, which is rigid and sequential, XP adapts easily to changing user requirements. Compared to Spiral, XP is simpler and more practical for small-to-medium teams. While Prototype focuses mainly on early UI design, XP delivers fully functional, tested increments throughout development. Its emphasis on stakeholder collaboration, high-quality code, andfrequent releases ensures that the system meets real-world needs efficiently and reliably, making it the best choice for this project.

# 1.3 Project Role Identification and Responsibilities

**GitHub Master:**

* **Nuha**: Manages the GitHub repository, merges contributions, organizes team tasks, and oversees project progress.

**Contributors:**

* **Rezowan, Maoun, Soumik**: Upload work, implement assigned features, assist with testing, and submit code to GitHub.

### Requirements Gathering

* **Nuha**: PRD, project background, non-functional requirements, software metrics.
* **Rezowan**: Functional requirements, project roles.
* **Maoun**: Process models, software metrics.
* **Soumik**: Scheduling, blood donation workflow analysis.

### Design Phase

* **Rezowan**: Use case diagram, activity diagram, data flow diagrams, donor profile UIs.
* **Maoun**: Class diagram, Admin dashboard UI.
* **Soumik**: Class diagram, Hospital/Blood Bank dashboard UI.
* **Rezowan**: Donor dashboard UI.
* **Nuha**: Login page and AI chatbot UI.

### Implementation

* All members coded their assigned modules and uploaded them to GitHub.

### Testing

* **Nuha**: Login, donation appointment booking, notices, AI chatbot.
* **Rezowan**: Fingerprint login, forget password, profile editing, notifications.
* **Maoun**: Donor/staff management, blood inventory, donation records, hospital database.
* **Soumik**: Blood request analysis, scheduling, alerts, reports.

### Decision-Making & Quality Control

* **Decision-making**: Maoun and Rezowan
* **Quality checking**: Soumik with support from Rezowan
* **Resource & task management**: Nuha via GitHub and Trello

### Task Allocation

* **Nuha**: GitHub management, login system, AI chatbot.
* **Rezowan**: Functional analysis, diagrams, donor profile UIs.
* **Maoun**: Class diagrams, metrics, Admin dashboard, estimations.
* **Soumik**: Class diagrams, testing, scheduling, analytics.

**Conclusion:**

* Clear task distribution ensured smooth coordination, timely completion, and a high-quality Blood Donation Management System.

## 2. SOFTWARE REQUIREMENTS SPECIFICATIONS (SRS) / PRODUCT REQUIREMENTS DOCUMENT (PRD)

## 2.1 Functional Requirements

* Provide a clear description of the major functionalities your system will perform.
* List down the core services, operations and features that users will be able to use.
* Explain how each function supports the overall objectives of the project.
* Describe workflows that represent how a user will complete a specific task through the system.
* Represent requirements using user stories or use case style descriptions, where appropriate.
* Define acceptance criteria that determine when a functional requirement is successfully implemented.

## 2.2 Non-Functional Requirements

* Identify the quality attributes that the system should satisfy in addition to its functional behaviour.
* **Performance**: What response times, processing capacity, or efficiency levels are expected?
* **Reliability**: How will the system ensure stable and uninterrupted service?
* **Integrity/Security**: What protections will safeguard data, authentication, authorization, and privacy?
* **Usability**: What level of ease of use, accessibility, and user experience should be maintained?
* **Maintainability**: How will the system support future modifications, bug fixes, or upgrades?
* **Scalability**: How should the system grow to support more users, data, or extended features?

# 3. PROJECT ESTIMATION AND SCHEDULING

## 3.1 Effort and Cost Estimation

* Define the scope of the project clearly before estimation.
* Apply **Lines of Code (LOC)** estimation by predicting the size of the system and converting it into effort using productivity rates.
* Apply the **COCOMO model** by selecting the appropriate project mode and calculating effort, development time, and staffing.
* Present results from all three estimation methods to show effort in person-hours or person-months.
* Mention assumptions and possible variation in results, since estimation always involves some level of uncertainty.

## 3.2 Project Scheduling

* Break the project into smaller tasks, show how the tasks are connected, and assign responsibilities.
* Allocate effort across phases such as analysis/design, coding, and testing (for example, using the 40–20–40 guideline).
* Develop a schedule with milestones and deadlines, represented in a timeline or Gantt chart.
* Include major deliverables tied to milestones and ensure outputs are clearly defined.
* Track progress through regular meetings, milestone reviews, and schedule checks.
* Use **Earned Value Analysis (EVA)** with measures such as Planned Value (BCWS), Earned Value (BCWP), Actual Cost (ACWP), Schedule Performance Index (SPI), Cost Performance Index (CPI), and variances (SV, CV) to evaluate progress.
* Recognize common causes of delay such as unrealistic deadlines, changing requirements, risks, technical or human issues, and miscommunication, and plan accordingly.

# SOFTWARE DESIGN

**4.1 System Design**

* Draw the system design for your project using **Draw.io** or **LucidChart**.
* Prepare a **Use Case Diagram** by first defining all users (actors) and their roles. Show each actor’s interactions with the system through use cases inside a system boundary. Include relationships like include or extend where needed.
* Prepare a **Class Diagram** by identifying the main classes from your project. Add attributes and operations for each class, and show associations, generalizations, aggregations, or compositions between them.
* Prepare an Activity Diagram that visually represents the workflow of a system or process.
* Prepare a **Data Flow Diagram (DFD)** starting with the **Context Diagram** (system as one process with external entities).
* Ensure consistency in naming actors, classes, processes,and data flows across all diagrams so they support each other.

## UI / Wireframe Design

* Design the user interface of your project individually using **Figma** or **Balsamiq**.
* Build the wireframes directly from your **PRD table**. Each functional requirement defined in the PRD should have a corresponding screen or component in the UI.
* Use the **Trello user board** (or equivalent task board) as a reference to decide which features and user flows need to be represented in the prototype.
* Create clickable wireframes that connect the main screens together, showing how a user will navigate through the system.
* Ensure the wireframe matches the actors, roles, and functionalities described in the PRD and system design diagrams.
* Export the prototype and include screenshots or a link to your design as part of the report submission.

# GIT WORKFLOW

* Create a central repository for the project on GitHub and set the **master (or main) branch** as the primary branch for integration.
* Each member should clone the repository and create their own **feature branches** for assigned tasks. Work on new features or fixes within these branches.
* Add files, stage them and commit changes with clear messages that describe the purpose of each update.
* Push commits from the feature branches to the remote repository so other members can see progress.
* Use **pull** to fetch and integrate changes from the remote repository into local copies, ensuring everyone stays updated.
* Merge feature branches into the **master/main branch** only after the work is tested and reviewed, resolving any conflicts that occur.
* Show evidence of collaboration by maintaining a clear commit history (using logs) with multiple commits, merges and contributions from all group members.
* Keep the repository organized with a clean history that tracks the project workflow from initialization to completion.

# TESTING

* The goal is to show how testing ensures quality and requirements conformance.
* Identify some testing methods that you want to use in the testing phase later for your project.
* Prepare **test cases** using a manual test case template which template taught you in the class.

# SOFTWARE PRODUCT METRICS

* Apply software product metrics to evaluate your project.
* Use **Function-based Metrics (Function Points)** by identifying inputs, outputs, files, interfaces, and inquiries to measure the functionality delivered by the system.
* Apply **Object-Oriented and Class Metrics** such as size, complexity, coupling, cohesion, and inheritance measures (e.g., WMC, DIT, NOC, CBC, LCOM) to assess the structure and quality of your design.
* Include **Operation-oriented Metrics** by measuring average operation size, complexity, and number of parameters per operation.
* Apply **Maintenance Metrics** such as the **Software Maturity Index (SMI)** to check the stability of your project across changes and updates.

# CONCLUSION AND FUTURE WORK

**Conclusion:**  
The Blood Donation Management System efficiently manages donors, hospitals, and blood banks, streamlining registration, scheduling, and inventory management. Using the XP model ensured adaptability, collaboration, and a reliable, user-friendly system.

**Future Work:**

1. Develop a mobile app for donors and hospitals for easier access and real-time notifications.
2. Integrate AI-based donor matching based on urgency, location, and blood type.
3. Use predictive analytics to forecast blood shortages and optimize donation drives.
4. Connect with national health systems to improve coordination during emergencies.
5. Enhance security with multi-factor authentication and data encryption.
6. Generate automated reports and alerts for hospitals and donors in critical situations.

**Text Format:**

* Style: Times New Roman
* Size: 12
* Space: 1.15
* Alignment: Justify