

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

**PROJECT TITLE**

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 |  |  |  |  |
| 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.

## Solution to the Problem and Process Model Selection

* Describe what your project scopes and features are.
* 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.

# 1.3 Project Role Identification and Responsibilities

* Identify the main roles involved in the project, covering both development and management aspects.
* Describe the responsibility of each role in key stages such as requirements gathering, design, implementation, testing and deployment.
* Specify which roles are responsible for decision making, quality assurance and resource management.
* Explain how responsibilities are distributed among the team members and justify the allocation based on skills and expertise.

## 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.

BloodLink is a web application for Blood Donation Management System. This application’s goal is to create a user friendly system that will simplify the whole process of blood donation. The system helps Blood donors, blood bank staff, hospital staff, delivery personnel and administrators with features like registration, blood inventory, delivery tracking, delivery assignment and management.

* Apply **Lines of Code (LOC)** estimation by predicting the size of the system and converting it into effort using productivity rates.

We are assuming that our project will need 85000 lines of code. Then if a developer writes 50 lines of code a day.

It is about 57 person-months.

* Apply the **COCOMO model** by selecting the appropriate project mode and calculating effort, development time, and staffing.

SLOC = 85000 (approximate)  
let’s assume this is organic type software project  
Coefficient<Effort Factor> = 2.4  
P = 1.05  
T = 0.38

Effort = PM = Coefficient <Effort Factor> \* (SLOC/1000)^P  
 = 2.4 \* (85000/1000)^1.05  
 = 254.74

Development time = DM = 2.50 \* (PM)^T  
 = 2.50 \* (254.74)^0.38  
 = 20.52

Required people = ST = PM / DM  
 = 254.74 / 20.52  
 = 12.41 ~ 13

* Effort: about 254.74 person-months
* Development Time: about 20.52 months
* Team Size: about 13 people
* Present results from all three estimation methods to show effort in person-hours or person-months.

Lets use SLIM model

LOC = 85000  
B = 2.5  
P = 800 (typically)  
Lets assume we will finish in 20 months

So, Effort ,

E = 18.57 person-months

By first effort estimation we got 57 person-months  
By Cocomo model we got 254.74 person-months  
By Slim model we got 18.57 person-months

* Mention assumptions and possible variation in results, since estimation always involves some level of uncertainty.  
  First we assumed that our project is 85000 lines of code. Then in first estimation we thought that a developer can write 50 lines of code in a day. In slim model we assumed productivity factor as 2.5 and productivity parameters 800. And finally project time 20 months.

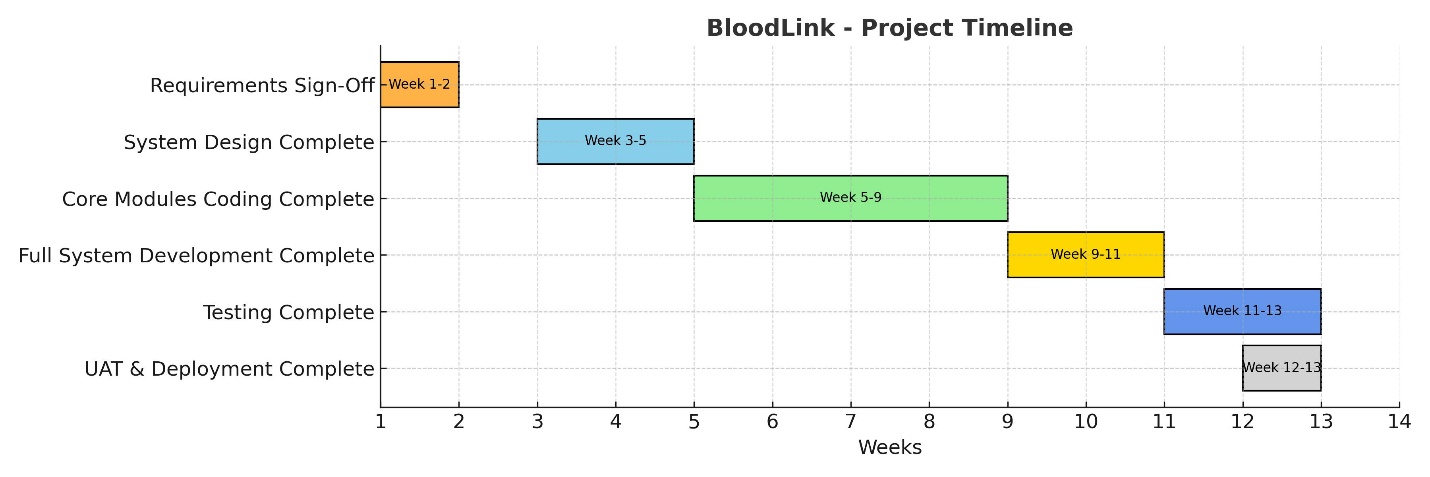
## 3.2 Project Scheduling

* Break the project into smaller tasks, show how the tasks are connected, and assign responsibilities.

Task Breakdown :  
  
1. **Requirement Analysis:** Collect functional & non-functional requirements, stakeholder interviews & surveys, define user roles & permissions  
2. **System Design:** Design database schema for donors, hospitals, blood units, UI/UX design for donor portal, staff dashboard, design APIs for hospital requests & logistics updates  
3. **Implementation:** Coding and building whole system  
4. **Testing:** Unit Testing, Integration Testing, System Testing  
5. **Deployment & Documentation:** Deploy the system, staff training & documentation  
6. **Maintenance & Future Improvements:** Bug fixing & updates, Monitor performance & reliability, Add new features  
  
Task Dependency:  
Requirement Analysis has no dependency  
System Design depends on requirements analysis  
Implemention depends on System Design  
Testing comes after implementation  
After testing System is deployed

* Allocate effort across phases such as analysis/design, coding, and testing (for example, using the 40–20–40 guideline).

|  |  |  |
| --- | --- | --- |
| Phase | % Effort | Tasks Included |
| Analysis & Design | 40% | Requirement analysis, UI/UX design, database/API design |
| Coding / Implementation | 20% | Development of portals, dashboards, logistics & admin modules |
| Testing & Deployment | 40% | Unit & integration testing, UAT, deployment, training |

* 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.

Major deliverables

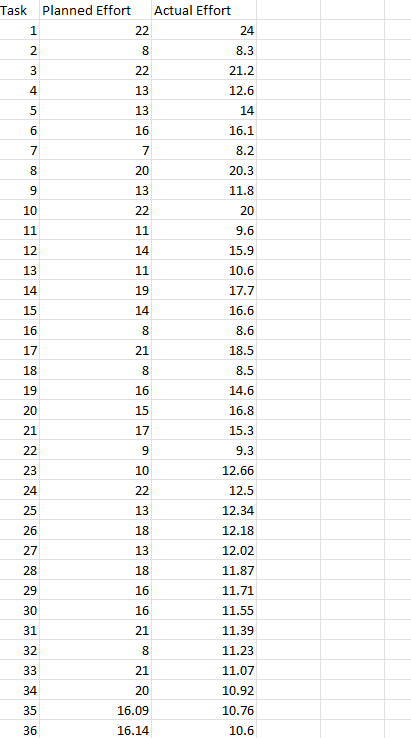
* Requirement Analysis : Requirement documents (User stories & roles)
* Design Complete : Design documents (ER diagram & Wiireframes)
* Core Modules Coding : Donor & Blood Bank Modules
* System Complete : Full System (Integrated system for all users)
* Testing Complete : QA Reports
* Deployment Complete : Live system
* Track progress through regular meetings, milestone reviews, and schedule checks.

 **Regular Meetings:** Weekly progress updates with team leads.

 **Milestone Reviews:** After design, coding, testing, and deployment.

 **Schedule Checks:** Compare actual progress vs planned milestones using simple % complete.

* 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.



A screenshot of a graph

AI-generated content may be incorrect.

BAC (Budget at Completion) = 2708.49  
If we assume 45 task is done from 70 scheduled task  
BCWP (Budgeted Cost of Work Performed) = 694.96  
BCWS (Budgeted Cost of Work Scheduled) = 1128.25  
ACWP (Actual Cost of Work Performed) = 569.57  
Schedule performance Index, SPI = (BCWP/BCWS) = 0.616  
Schedule variance, SV = (BCWP – BCWS) = -433.29  
% Scheduled = (BCWS/BAC) = 0.4166  
% Completed = (BCWP/BAC) = 0.2566  
Cost Performance Index, CPI = (BCWP/ACWP) = 1.2201  
Cost Variance, CV = BCWP – ACWP = 125.39

* Recognize common causes of delay such as unrealistic deadlines, changing requirements, risks, technical or human issues, and miscommunication, and plan accordingly.

|  |  |
| --- | --- |
| Cause | Mitigation Plan |
| Unrealistic deadlines | Multiple estimation methods |
| Changing requirements | Use change control process, approve before implementation |
| Risks (technical/human) | Risk register, contingency planning |
| Miscommunication | Regular meetings, clear documentation, centralized tracking |
| Technical issues | Prototype critical components, code reviews |
| Staff availability | Cross-train team, backup assignments |

# 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.   
    
  **Size**
* Number of classes: 11  
  (User, Donor, BloodBankStaff, HospitalStaff, Administrator, Volunteer, Campaign, Appointment, Inventory, Delivery, Requests)
* Number of modules (features): 36

**Complexity**

Classes are interconnected with multiple other classes. For example:

* + Donor connects with Appointment, User, and Campaign.
  + BloodBankStaff connects with Inventory, HospitalStaff, and indirectly Donor.
  + Delivery connects with Inventory, HospitalStaff, and Administrator

**Coupling**

Classes and their coupled classes:

* + User - Donor, BloodBankStaff, HospitalStaff, Administrator, Campaign, Volunteer
  + Donor - User, Appointment, Campaign
  + BloodBankStaff - User, Donor, HospitalStaff, Inventory
  + HospitalStaff - BloodBankStaff, Inventory, Delivery
  + Administrator - User, Inventory, Delivery
  + Campaign - User, Donor, Appointment
  + Volunteer - User, Appointment
  + Appointment - Donor, Volunteer, Campaign
  + BloodUnit - BloodBankStaff, HospitalStaff, Delivery, Administrator
  + Delivery - Inventory, HospitalStaff, Administrator
  + Requests – User, HospitalStaff, Administrator

**Cohesion (LCOM values approximation)**

* + User: 3
  + Donor: 3
  + BloodBankStaff: 3
  + HospitalStaff: 3
  + Administrator: 3
  + Volunteer: 2
  + Campaign: 2
  + Appointment: 0
  + Inventory: 2
  + Delivery: 2
  + Requests: 3

**Sufficiency**

* + Most classes (Donor, Appointment, Inventory, Delivery) cover their intended responsibilities.
  + User acts as a base identity class, ensuring role-based access.
  + Administrator and BloodBankStaff have slightly overlapping responsibilities (inventory, reporting).

**Completeness**

* + Domain entities (Inventory, Appointment, Delivery) are reusable.
  + Role-based classes (Donor, HospitalStaff, Volunteer) are specific to the system.
  + Admin-related tasks are less reusable due to role concentration.

**Primitiveness**

* + Most operations are atomic (e.g., manageDonors(), requestBlood(), assignDelivery()).

**Similarity**

* + User-type classes: Donor, BloodBankStaff, HospitalStaff, Volunteer, Administrator share login/role-related behaviors.
  + Domain-entity classes: Inventory, Appointment, Delivery are persistent entities with CRUD-like operations.
  + Role-specialization classes: Staff roles and organizer/volunteer extend the system with specialized responsibilities.
* Use **Function-based Metrics (Function Points)** by identifying inputs, outputs, files, interfaces, and inquiries to measure the functionality delivered by the system.

**External Inputs (EIs):**

* + User registration/login/logout
  + Update profile
  + Book appointment (donor)
  + Schedule appointment (staff)
  + Manage donors (staff)
  + Manage inventory (staff)
  + Request blood (hospital staff)
  + Assign delivery (admin)
  + Update delivery status (delivery)
  + Plan/promote drive (organizer)
  + Volunteer shift management

**External Outputs (EOs):**

* + View donor history
  + View eligibility
  + View hospital request status
  + View appointment list
  + Generate reports (admin, staff)
  + Track delivery status
  + Campaign/drive report
  + Notifications/alerts

**Internal Logical Files (ILFs):**

* + User records
  + Donor records
  + Appointment records
  + Blood inventory
  + Delivery records
  + Campaign records

**External Interface Files (EIFs):**

* + Notification system (alerts/email/SMS)
  + Hospital integration

**External Inquiries (EQs):**

* + View appointment status
  + Track request status
  + View donor records
  + View delivery assignment
  + View campaign performance

**Function Point Calculation:**

* + EIs: 12 × 4 = 48
  + EOs: 8 × 5 = 40
  + ILFs: 6 × 10 = 60
  + EIFs: 2 × 7 = 14
  + EQs: 5 × 4 = 20

Total Function Points = 182

* 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.  
    
  **Donor Class**
  + Methods: 3
  + WMC: 3
  + DIT: 1
  + NOC: 0
  + CBC: 3 (User, Appointment, Campaign)
  + LCOM: 3

**Appointment Class**

* + Methods: 0
  + WMC: 0
  + DIT: 0
  + NOC: 0
  + CBC: 3 (Donor, Volunteer, Campaign)
  + LCOM: 0

BloodBankStaff Class

* + Methods: 3
  + WMC: 3
  + DIT: 1
  + NOC: 0
  + CBC: 3 (Donor, Inventory, HospitalStaff)
  + LCOM: 3

**HospitalStaff Class**

* + Methods: 2
  + WMC: 2
  + DIT: 1
  + NOC: 0
  + CBC: 3 (BloodBankStaff, Inventory, Delivery)
  + LCOM: 3

**Administrator Class**

* + Methods: 3
  + WMC: 2
  + DIT: 1
  + NOC: 0
  + CBC: 3 (User, Inventory, Delivery)
  + LCOM: 3

**Volunteer Class**

* + Methods: 2
  + WMC: 2
  + DIT: 1
  + NOC: 0
  + CBC: 2 (User, Appointment)
  + LCOM: 2

**Campaign Class**

* + Methods: 2
  + WMC: 2
  + DIT: 1
  + NOC: 0
  + CBC: 2 (User, Donor, Appointment)
  + LCOM: 2

**Inventory Class**

* + Methods: 2
  + WMC: 2
  + DIT: 0
  + NOC: 0
  + CBC: 4 (BloodBankStaff, HospitalStaff, Delivery, Admin)
  + LCOM: 2

**Delivery Class**

* + Methods: 2
  + WMC: 2
  + DIT: 0
  + NOC: 0
  + CBC: 3 (BloodUnit, HospitalStaff, Administrator)
  + LCOM: 2
* Include **Operation-oriented Metrics** by measuring average operation size, complexity, and number of parameters per operation.

Total methods/operations: **26**  
Average operations per class: **2.36**  
Parameters per method: 1.11

* Apply **Maintenance Metrics** such as the **Software Maturity Index (SMI)** to check the stability of your project across changes and updates.

Total modules, MT = 36  
added modules, Fa = 3  
Changed modules, Fc = 6  
Deleted modules, Fd = 1

SMI is not 1 but it is little close to it.

# CONCLUSION AND FUTURE WORK

* Write a conclusion and future work based on your project.

**Text Format:**

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