Project Documentation Guidelines

Item	Student Deadline	Graduate Deadline
Project Planning & Management	9/21/2025	9/21/2025
Literature Review	9/21/2025	9/21/2025
Requirements Gathering	9/21/2025	9/21/2025
System Analysis & Design	9/21/2025	9/21/2025
Implementation (Source Code & Execution)	11/15/2025	11/15/2025
Final Presentation & Testing& Reports	11/20/2025	11/20/2025

***** All documents should be uploaded to GitHub.**

1. Project Planning & Management

- **Project Proposal** – Overview of the project, objectives, and scope.
- **Project Plan** – Timeline (Gantt chart), milestones, deliverables, and resource allocation.
- **Task Assignment & Roles** – Defined responsibilities for team members.
- **Risk Assessment & Mitigation Plan** – Identifying risks and solutions.
- **KPIs (Key Performance Indicators)** – Metrics for project success (e.g., response time, system uptime, user adoption rate).

2. Literature Review

- **Feedback & Evaluation** – Lecturer’s assessment of the project.
- **Suggested Improvements** – Areas where the project can be enhanced.
- **Final Grading Criteria** – Breakdown of marks based on documentation, implementation, testing, and presentation.

3. Requirements Gathering

- **Stakeholder Analysis** – Identifying key stakeholders and their needs.
- **User Stories & Use Cases** – Scenarios illustrating how users interact with the system.
- **Functional Requirements** – List of features and functionalities.
- **Non-functional Requirements** – Performance, security, usability, and reliability criteria.

4. System Analysis & Design

i. **Problem Statement & Objectives – Define the problem being solved and project goals.**

• Use Case Diagram & Descriptions – Identify system actors and interactions.

• Functional & Non-Functional Requirements – Clearly state system capabilities and constraints.

• Software Architecture – High-level design outlining system components, interactions, and architecture style (e.g., MVC, Microservices).

ii. **Database Design & Data Modeling**

• ER Diagram (Entity-Relationship Diagram) – A well-defined ERD showcasing database structure and relationships.

• Logical & Physical Schema – Tables, attributes, keys, and normalization considerations.

iii. **Data Flow & System Behavior**

• DFD (Data Flow Diagram) – Context-level and detailed levels showing how data moves through the system.

• Sequence Diagrams – Process flow representation of key interactions between components.

• Activity Diagram – Visualizing the workflow of processes or user actions within the system.

• State Diagram – Represents different states of an object and how it transitions between them.

• Class Diagram – Defines the structure of the system by showing classes, attributes, methods, and relationships.

iv. UI/UX Design & Prototyping

• Wireframes & Mockups – Screens and visual representations of the user interface.

• UI/UX Guidelines – Design principles, color schemes, typography, and accessibility considerations.

v. System Deployment & Integration

• Technology Stack – Backend, frontend, and database technologies.

• Deployment Diagram – Describes how software components are distributed across hardware.

• Component Diagram – Shows high-level system components and their dependencies.

vi. Additional Deliverables (if applicable)

• API Documentation – If the system includes APIs, provide documentation for endpoints and usage.

• Testing & Validation – Unit tests, integration tests, and user acceptance testing plan.

• Deployment Strategy – Hosting environment, deployment pipelines, and scaling considerations.

5. Implementation (Source Code & Execution)

1. Source Code

• Structured & Well-Commented Code – Clean, maintainable, and properly documented code following best practices.

• Coding Standards & Naming Conventions – Consistent formatting, meaningful variable names, and adherence to industry standards.

• Modular Code & Reusability – Organized into reusable components, functions, and classes.

• Security & Error Handling – Secure coding practices, validation checks, and proper exception handling.

2. Version Control & Collaboration

• Version Control Repository – Hosted on GitHub, GitLab, or Bitbucket with a public/private repository link.

• Branching Strategy – Clear workflow (e.g., GitFlow, Feature Branching) for managing code updates.

• Commit History & Documentation – Meaningful commit messages and detailed pull request descriptions.

• CI/CD Integration (if applicable) – Automated builds, testing, and deployment pipelines.

3. Deployment & Execution

- **README File – Includes:**
- **Installation steps**
- **System requirements (hardware/software dependencies)**
- **Configuration instructions**
- **Execution guide (running the project locally or accessing a deployed version)**
- **API documentation (if applicable)**
- **Executable Files & Deployment Link –**
- **Compiled software or packaged application (e.g., .exe, .jar, .apk).**
- **Deployed web/mobile app**

6. Testing & Quality Assurance

- **Test Cases & Test Plan** – Document detailing test scenarios and expected outcomes.
- **Automated Testing (if applicable)** – Any automated test scripts used.
- **Bug Reports** – Issues identified and resolutions.

7. Final Presentation & Reports

- **User Manual** – Instructions for end users.
- **Technical Documentation** – System architecture, database schema, API documentation.
- **Project Presentation (PPT/PDF)** – Summary of the project, challenges, solutions, and outcomes.
- **Video Demonstration (Optional)** – Short demo showcasing the project’s functionality.