MGM University

Institute of information and communication Technology IICT

Guidelines for Project Monitoring

Project Monitoring-I

Students need to present project work as per given guidelines

1. Literature Review & Background Study

- Conduct research on existing solutions or systems.
- Study and Review **research papers**, **journals**, **technical blogs**, **theses**, and **project reports** from credible sources (IEEE, Springer, Elsevier, arXiv, etc.).
- Compare existing models or systems related to your topic.
- Study and Analyze tools, technology frameworks, algorithms, and datasets commonly
- Understand the **advantages and limitations** of current technologies.
- Summarize findings and highlight the **research gap** your project aims to address.
- Define the problem statement clearly.
- Define objectives of the project

2. Requirement Analysis

a. Functional Requirements

These define *what* the system or software is expected to do. Examples:

- User login/authentication.
- Data input and validation.
- Report generation.
- Recommendation based on user inputs (in AI/ML).
- CRUD operations for a database.

b. Technical Requirements:

- Programming languages (e.g., Python, Java, JavaScript).
- Frameworks/libraries (e.g., TensorFlow, Django, React).
- Database (e.g., MySQL, MongoDB).
- Tools (e.g., QGIS, Git, VS Code).
- Hardware needs (e.g., GPU for ML training).
- APIs or third-party services.

3. System Design

The **Design** phase translates the requirements into a blueprint for building your system. It defines how the system will be structured, how components interact, and how users will interact with it.

- System design Diagram.
- UML diagrams (Use Case, DFDs, Class diagrams, etc.).
- Database schema and ER diagram.
- UI/UX wireframes or screen sketches.

- List of technologies used

1. System design A high-level structure showing major components and their interactions.
- **Monolithic**, **client-server**, or **microservices** architecture.
- Example: For a web-based AI system: Frontend → API Layer → ML Model → Database.

2. Module Design Break down the system into modules or subsystems.
- Each module should represent a major function (e.g., user management, data preprocessing, model training).
- Define input/output for each module.

3. Data Flow Diagrams (DFD) Shows how data moves through the system.
- Levels: Context Level (0), Level 1, Level 2 DFDs.
- Useful for understanding interactions between users, processes, and data stores.

4. UML Diagrams *(Unified Modeling Language)* – Optional but useful
- **Use Case Diagram**: Shows user-system interactions.
- **Class Diagram**: For object-oriented design.
- **Sequence Diagram**: For workflow or timing between objects/modules.
- **Activity Diagram**: For processes and decisions.

5. Database Design
- **ER Diagram (Entity Relationship)**: Shows entities and relationships.
- Define tables, fields, keys, constraints.
- Normalize the schema if needed.

6. UI/UX Design
- Design basic **wireframes** or mockups for user interfaces.
- Ensure intuitive navigation and good usability.
- Use tools like Figma, Canva, Adobe XD, or even hand-drawn sketches.

Specify tools, programming languages, and frameworks used:

- **Frontend**: HTML, CSS, JS, React, etc.
- **Backend**: Node.js, Django, Flask, etc.
- **AI/ML**: Python, TensorFlow, Scikit-learn, etc.
- **Database**: MySQL, Firebase, MongoDB

.

4 Implementation / Development (20%)

- The **Implementation** (or **Development**) phase is where the system design is converted into a working solution. This stage involves writing code, integrating components, and building the actual product or prototype based on the project plan.

    - Start coding or system construction phase.
    - Follow best coding practices and version control (e.g., Git).
    - Integrate modules iteratively (Agile/Scrum approach if applicable).

1 Environment Setup

- Install necessary tools, libraries, and dependencies.
- Set up development environments (e.g., IDEs like VS Code, PyCharm, Eclipse).
- Configure databases, servers, or cloud platforms (if applicable).

2. Module-wise Coding

- Break down the project into **small, manageable modules**.
- Follow the **design documents** for logic and structure.
- Implement **backend**, **frontend**, and **middleware** components (as per tech stack).

3. Use of Frameworks and Libraries

- AI/ML Projects: TensorFlow, Keras, PyTorch, Scikit-learn.
- Web Development: React, Angular, Django, Flask, Node.js.
- Mobile Apps: Flutter, Android SDK.
- Databases: MySQL, MongoDB, Firebase.

4. Integration

- Combine modules and ensure smooth data flow between components.
- For ML projects: integrate data preprocessing → model training → result visualization.
- Use **APIs** or **microservices** where needed.

Tools & Technologies (Examples):

Area	Tools/Languages/Frameworks
Programming	Python, Java, JavaScript, C#
AI/ML	TensorFlow, Scikit-learn, Pandas, NumPy
Web Dev	HTML/CSS, JS, React, Node.js, Django
Database	MySQL, SQLite, MongoDB, Firebase
IDEs	VS Code, PyCharm, Eclipse, Android Studio
Version Control	Git, GitHub, Bitbucket

Project Monitoring-II

Implementation / Development (80%)

Project Monitoring-III

Implementation / Development (100%)

Testing

- Perform unit, integration, and system testing.
- Identify bugs and optimize performance.
- Validate output with test data and real-world scenarios.

Documentation

- Prepare detailed project report including:
  - Introduction, Objectives
  - Literature Review
  - Methodology / Design
  - Results and Discussion
  - Conclusion and Future Scope
- Maintain a logbook (if required).

Final Submission

- Submit report, code, and related files to your department.
- Upload to GitHub or institutional repository (if required).
- Prepare Publish and submit research paper on project work
- Submit certificate of completions, hackathons (if participated)