Day 3

Software Development & Project Planning

Minor vs Major Project

Workshop By:

Dr. Vishal Bharvesh

Ishanvi
Info-Soft Pvt. Ltd.

Understanding Projects

1. **Every project starts with a problem:** Every project begins by identifying a real-world problem or need. This problem gives direction and purpose to the project. Without a clearly defined problem, development becomes confusing, and the final outcome may not be useful or meaningful.

2. **Solution must be clear and complete:** The solution developed in a project should be well-defined, understandable, and fully functional. It should cover all required features within its scope, include proper validation and security, and should not leave any important part incomplete or unclear.

Minor Project

1. **Small scope:** A minor project has a limited and well-defined scope. It focuses on a specific functionality instead of covering the entire system. This makes it easier to design, develop, test, and complete within a short time period effectively.

2. **Focused module:** A minor project usually concentrates on one specific module of a larger system. It ensures that the selected module is deeply understood and properly implemented, rather than trying to handle multiple modules with incomplete or unclear functionality.

3. **Complete solution:** Even though the project is small, it must provide a complete and working solution. All features within its scope should be properly implemented, tested, and validated, without leaving any part unfinished or partially developed.

4. **Reusable in future:** A well-designed minor project can be reused as a module in a major project later. This helps in saving time, improving efficiency, and building a strong foundation for developing more complex and larger systems in the future.

Major Project

1. **Full system :** A major project represents a complete system that covers all functionalities required to solve a real-world problem. It integrates different components and ensures that the entire workflow, from input to output, is properly designed and executed.

2. **Multiple modules:** A major project is divided into multiple modules, each handling a specific function. These modules work together in an integrated manner, ensuring smooth communication and data flow across the system to achieve overall functionality and performance.

3. **Real-world complexity:** Major projects deal with real-world scenarios that involve multiple users, conditions, and exceptions. They require handling complex logic, dynamic data, and various situations, making the system more practical, realistic, and useful in real-life environments.

4. **Detailed implementation:** In a major project, every component is implemented in detail, including database design, frontend, backend logic, validation, and security. Each module is thoroughly developed and tested to ensure accuracy, reliability, and overall system efficiency.

Key Principle

1. **Do not leave project incomplete:** A project should never be left partially developed or unfinished. Even if the scope is small, all planned features must be properly implemented, tested, and working. An incomplete project creates confusion and fails to demonstrate real understanding.

2. **Depth may vary but completeness is must:** The level of detail or depth in a project can vary depending on whether it is minor or major. However, within the selected scope, the solution must always be complete, functional, and properly validated without missing any essential components.

Example: Inventory System

1. **Minor: Inventory module only:** In a minor project, the focus is only on the inventory module. It includes functions like adding items, updating stock, deleting records, and tracking quantities, providing a complete solution for inventory management within a limited scope.

2. **Major: Full store system**: In a major project, the entire store system is developed, including inventory, billing, customer management, and admin control. All modules are integrated to handle complete operations, reflecting real-world business processes with detailed functionality and interactions.

Development Layers

1. **Frontend:** Frontend is the user interface of the application where users interact with the system. It includes forms, buttons, and visual elements. A good frontend should be user-friendly, responsive, and capable of collecting correct input from users.

2. **Backend:** Backend is the logic layer of the application that processes user requests. It handles business logic, validations, and communication between frontend and database. Backend ensures that the system works correctly and efficiently behind the scenes.

3. **Database:** Database is used to store and manage all application data in an organized manner. It includes tables, relationships, and constraints. A well-designed database ensures data accuracy, consistency, and easy retrieval whenever required by the system.

4. **Integration:** Integration connects frontend, backend, and database into a single working system. It ensures smooth data flow and communication between all components. Proper integration makes the application functional, reliable, and capable of performing complete operations without errors.

Validation & Security

1. **Input validation:** Input validation ensures that the data entered by users is correct, complete, and in the expected format. It prevents invalid or harmful inputs, reduces errors, and improves system reliability by checking data before processing or storing it.

2. **Secure database:** A secure database protects stored data from unauthorized access, modification, or loss. It includes techniques like authentication, encryption, and access control. Database security ensures confidentiality, integrity, and safety of sensitive information within the system.

3. **Error handling:** Error handling manages unexpected issues during system execution. It ensures that errors are properly detected, controlled, and communicated to the user in a clear way. Good error handling prevents system crashes and improves user experience and system stability.

Documentation

1. **Clear explanation:** Clear explanation means the project should be properly described in simple and understandable language. It should explain the problem statement, objectives, methodology, tools used, and expected outcomes so that anyone reading the documentation can easily understand the system.

2. **Diagrams:** Diagrams visually represent the system structure and workflow. They include ER diagrams, flowcharts, and sequence diagrams. Diagrams make complex concepts easier to understand and help evaluators quickly see how different components of the project are connected.

3. **Code inclusion:** Code inclusion means adding important source code in the project documentation. It demonstrates actual implementation and logic used in development. Properly formatted and explained code helps evaluators understand functionality and proves that the project is practically implemented.

Diagrams

1. **ER Diagram:** An ER Diagram (Entity Relationship Diagram) represents the structure of the database. It shows entities, attributes, and relationships between them. It helps in designing a well-organized database and understanding how different data elements are connected.

2. **Flowchart:** A flowchart shows the step-by-step flow of a process in a system. It uses symbols to represent actions, decisions, and flow direction. It helps in understanding the logic of the system and simplifies complex processes visually.

3. **Sequence Diagram:** A sequence diagram shows how different components or objects interact with each other over time. It represents the sequence of messages exchanged between elements, helping to understand system behavior and communication flow during execution.

Diagrams

4. **Gantt Chart:** A Gantt chart is used for project planning and scheduling. It shows tasks, timelines, and dependencies between activities. It helps in tracking progress, managing deadlines, and ensuring that the project is completed within the planned time frame.

5. **UML Diagram:** UML (Unified Modeling Language) diagrams represent the design and structure of a system. They include different types like class, use-case, and activity diagrams, helping developers visualize system architecture and improve communication and design clarity.

Minor Documentation

1. Cover Page (1 Page)
Project Title
Student Name(s)
Enrollment Number
Branch & Semester
College Name
Submission Year

2. Certificate (1 Page)
Signed by Guide / HOD
Project completion declaration

3. Declaration (1 Page)
Student declaration of originality

Minor Documentation

4. Acknowledgement (1 Page)
Thanks to guide, college, contributors

5. Abstract (1 Page)
Short summary of project
Problem + solution + outcome

6. Table of Contents (1–2 Pages)
Chapters with page numbers

Minor Documentation

MAIN CONTENT

7. Chapter 1: Introduction (4–6 Pages)
Background of problem
Objective of project
Scope
Purpose

8. Chapter 2: Literature Review (5–8 Pages)
Existing systems
Research or similar work
Limitations

9. Chapter 3: System Analysis (5–7 Pages)
Problem definition
Requirement analysis
Feasibility study

Minor Documentation

10. Chapter 4: System Design (8–12 Pages)
ER Diagram
Flowchart
UML Diagrams
Database design

11. Chapter 5: Implementation (10–15 Pages)
Tools & technologies
Frontend + Backend explanation
Screenshots

12. Chapter 6: Testing (4–6 Pages)
Test cases
Results
Error handling

13. Chapter 7: Results & Discussion (3–5 Pages)
Output explanation
System performance

Minor Documentation

14. Chapter 8: Conclusion & Future Scope (2–3 Pages)
Summary
Future improvements

15. References (2–3 Pages)
Books, websites, research papers

Appendix (Optional – 5–10 Pages)
Source code
Extra screenshots

Total Pages:

- **Minimum:** 60 pages
- **Ideal Range:** 70–80 pages

Major Documentation

1. Cover Page (1 Page)
Project Title
Student Name(s)
Enrollment Number
Branch & Semester
College Name
Academic Year

2. Certificate (1 Page)
Signed by Guide, HOD, Principal
Confirms project completion

3. Declaration (1 Page)
Student declaration of originality
No plagiarism statement

Major Documentation

4. Acknowledgement (1 Page)
Thanks to guide, institution, team

5. Abstract (1 Page)
Problem statement
Solution approach
Key results

6. Table of Contents (2–3 Pages)
Chapters with page numbers

7. List of Figures & Tables (1–2 Pages)
All diagrams, charts, tables

Major Documentation

MAIN CONTENT (DETAILED)

8. Chapter 1: Introduction (6–10 Pages)
Background of study
Problem statement
Objectives
Scope of system
Methodology overview

9. Chapter 2: Literature Review (10–15 Pages)
Existing systems
Research papers / technologies
Comparative analysis
Limitations of current systems

Major Documentation

10. Chapter 3: System Analysis (8–12 Pages)
Requirement analysis (Functional & Non-functional)
Feasibility study (Technical, Economic, Operational)
User analysis

11. Chapter 4: System Design (15–25 Pages)
Architecture design
ER Diagram (Detailed + Modular)
UML Diagrams:
- Use Case
- Class Diagram
- Sequence Diagram

Flowcharts
Database schema

Major Documentation

12. Chapter 5: Implementation (20–30 Pages)
Tools & technologies used
Module-wise explanation
Frontend + Backend
API / Integration logic
Screenshots of system

13. Chapter 6: Testing (8–12 Pages)
Test cases (Unit + Integration)
Test results
Bug handling
Validation checks

Major Documentation

14. Chapter 7: Results & Discussion (5–8 Pages)
Output analysis
Performance evaluation
Comparison with existing system

15. Chapter 8: Conclusion & Future Scope (3–5 Pages)
Summary of work
Achievements
Future enhancements

Major Documentation

16. References (3–5 Pages)
Books
Journals
Websites

17. Appendix (Optional – 10–20 Pages)
Source code
Sample data
Extra screenshots

Total Pages:

- **Minimum:** 120 pages
- **Ideal Range:** 150–200 pages

Problem Solving Focus

1. **Define problem clearly:** Clearly defining the problem means understanding exactly what issue the project is trying to solve. It should be specific, measurable, and relevant, so that the development process remains focused and the final solution directly addresses the identified need.

2. **Measure success:** Measuring success means evaluating how effectively the project solves the defined problem. This can be done using performance metrics, user feedback, accuracy, or efficiency. It helps determine whether the solution meets objectives and delivers expected results.

Minor Project

1. Smart Attendance System
Manual attendance is time-consuming and error-prone in institutions. This web-based system digitizes attendance, stores records securely, generates reports automatically, and provides role-based access for faculty and administrators.

2. Online Quiz Management System
Traditional exams require manual evaluation and delayed results. This platform conducts secure online quizzes, auto-grades responses instantly, maintains performance analytics, and generates leaderboards for transparent academic assessment.

3. Personal Finance Tracker
People struggle to track spending and savings effectively. This application records income and expenses, categorizes transactions, visualizes trends with charts, and generates monthly financial reports for smarter budgeting decisions.

Minor Project

4. Student Project Portal
Project submissions lack centralized tracking and feedback management. This portal enables students to upload projects, faculty to review submissions, provide feedback, approve work, and maintain structured academic project records.

5. AI Chatbot for College FAQs
Students repeatedly ask common institutional queries. This chatbot uses NLP techniques to understand questions and deliver instant, automated responses regarding admissions, schedules, facilities, and academic procedures.

6. Fake News Detection System
Misinformation spreads rapidly online, misleading users. This machine learning system analyzes text features, applies NLP classification algorithms, and predicts whether news articles are genuine or fake with accuracy metrics.

Minor Project

7. Handwritten Digit Recognition
Manual digit recognition from scanned forms is inefficient. This deep learning system uses convolutional neural networks to classify handwritten digits accurately, enabling automated form processing and validation.

8. Resume Screening System
Recruiters manually review numerous resumes inefficiently. This AI system compares resumes against job descriptions, extracts keywords, ranks candidates automatically, and streamlines the shortlisting process objectively.

Minor Project

9. Campus Navigation App
New students struggle locating campus facilities. This mobile application integrates maps and GPS, provides real-time directions, highlights departments and amenities, and improves campus navigation efficiency.

10. Daily Habit Tracker App
People fail maintaining consistent habits due to poor tracking. This app logs daily activities, tracks streaks, sends reminders, and visualizes progress to encourage discipline and goal achievement.

11. Blood Donor Finder App
Emergency patients face delays finding blood donors. This mobile application stores donor information, filters by blood group and location, and connects recipients with nearby volunteers instantly.

Minor Project

12. Password Strength Analyzer
Weak passwords increase cybersecurity risks. This tool evaluates password complexity using length and character diversity, estimates strength levels, and suggests improvements to enhance user security.

13. Ethical Keylogger Demonstration
Users remain unaware of keylogging threats. This educational project demonstrates keystroke logging techniques ethically, highlights vulnerabilities, and provides preventive cybersecurity recommendations against malicious spyware.

14. Network Traffic Monitor
Network misuse often goes undetected. This system monitors real-time traffic data, visualizes bandwidth usage, identifies suspicious activity patterns, and assists administrators in maintaining network security.

Major Project

1. AI-Based Disease Prediction System
Patients misinterpret symptoms, delaying diagnosis. This machine learning system analyzes symptom inputs, predicts possible diseases using trained models, and assists healthcare professionals with preliminary diagnostic insights.

2. Smart Traffic Management System
Urban congestion causes delays and fuel waste. This computer vision system analyzes vehicle density through cameras and dynamically adjusts traffic signals to optimize flow efficiency.

3. Face Recognition Attendance System
Manual attendance allows proxy entries and inefficiencies. This system captures facial images, applies recognition algorithms, verifies identities, and automatically records accurate attendance data.

Major Project

4. Stock Price Prediction Using LSTM
Investors struggle predicting volatile stock trends. This deep learning model uses historical time-series data with LSTM networks to forecast future price movements for informed decision-making.

5. E-Learning Platform with AI Recommendation
Students receive generic learning resources. This intelligent platform tracks performance, analyzes preferences, and recommends personalized courses and materials to enhance learning outcomes.

6. Online Voting System with Blockchain
Traditional voting risks tampering and lack of transparency. This blockchain-based system ensures secure vote recording, immutable transaction logs, and verifiable results maintaining electoral integrity.

Major Project

7. Cloud-Based File Storage System
Users require secure and accessible file storage. This system provides authenticated login, encrypted uploads, cloud storage management, file sharing, and retrieval across devices.

8. ERP System for Colleges
Institutions manage academic data inefficiently across departments. This ERP integrates attendance, fees, exams, and records into a unified platform improving administrative efficiency.

9. Intrusion Detection System
Cyberattacks often bypass traditional security systems. This machine learning-based IDS monitors network traffic, detects anomalous patterns, and alerts administrators about potential threats.

Major Project

10. Phishing Website Detection System
Users unknowingly access fraudulent websites. This classification system analyzes URLs and webpage features, predicts phishing attempts, and warns users before interaction.

11. Secure File Sharing with Encryption
File transfers risk interception and data breaches. This system encrypts files using AES/RSA algorithms, ensures secure transmission, and decrypts only for authorized recipients.

12. Smart Home Automation System
Manual appliance control wastes energy and time. This IoT system connects devices via sensors and microcontrollers, enabling remote monitoring and automated home appliance management.

13. IoT-Based Smart Agriculture System
Farmers face inefficient irrigation and crop monitoring. This system uses soil and climate sensors to automate watering, monitor conditions, and improve agricultural productivity.

Major Project

14. Smart Parking System
Drivers waste time searching parking spaces. This IoT-enabled system detects vacant slots using sensors and displays real-time availability through a mobile application.

15. Movie Recommendation System
Users struggle finding relevant movies. This recommendation system analyzes viewing history and preferences using collaborative filtering to suggest personalized movie options.

16. Crime Rate Prediction and Analysis
Law enforcement lacks predictive crime insights. This data analytics system examines historical crime data, identifies trends, and forecasts high-risk areas for preventive measures.

17. Customer Churn Prediction System
Businesses lose customers without understanding causes. This machine learning system analyzes behavioral data, predicts churn probability, and supports targeted retention strategies.

Final Thought

In any project, the true value does not come from how big it looks, but from how well it works.

A small project, if designed with clarity, accuracy, and proper implementation, can create a stronger impact than a large but incomplete system.

When you focus on quality, you pay attention to details, validation, user experience, and problem-solving effectiveness. This not only improves your technical skills but also builds confidence during presentation and viva.

Always remember, evaluators and real-world users appreciate a clean, working, and meaningful solution more than a complex but poorly executed project.