Assessment 2

Total Marks 50

Assignment: Software development, testing and configuration

Assignment Objective:

To simulate the real-world software development lifecycle by collaboratively designing, developing, and deploying an object-oriented application using advanced OOP concepts, popular design patterns, and GitHub for version control. The project will also involve a requirements analysis phase and UI/UX prototyping using Figma.

Task Overview:

Step	What to Do
Identify System Features	E.g., notifications, user roles, settings manager, third-party integrations
Make a Figma design of your software	Design Landing Page / Dashboard and Other Functional UI component
Map Features to Patterns	Use the matrix of the following
Select at least 5 Patterns	Pick ones that make sense for their solution
Connect Patterns with OOP	Shows how OOP and Design Patterns work together in the code
Team Collaboration via GitHub	Create a GitHub repository for the project and do teamwork for pushing, pulling, branching, and resolving conflict.
Document Usage	Make a complete report based on provided template

Task 1: Requirement Analysis:

- Identify a real-world problem your group wants to solve with a software system.
- Write a **Software Requirements Specification (SRS)** document that includes:
    - Problem Statement
    - Functional and Non-functional Requirements
    - System Overview diagram

Task 2: UI/UX Design Using Figma

- Based on the SRS, design the **User Interface (UI)** in Figma.
- Share the link to your peers and do collaboration (We will check individual contribution).
- Do prototyping on your design and **share the production level URL in report**

- Share the link to the Figma project in report.

Task 3: Implementation (Coding) Using Design Pattern and OOP Principles Using following Matrix.

3.1 Design Pattern Selection Matrix (Example usages)

Project Feature	Suggested Design Pattern	How to Use It
Integrating multiple sandbox payment gateways (e.g., Stripe, PayPal)	Adapter	Create a common interface to interact with different external services
Allowing users to customize dashboards with widgets	Decorator	Wrap base dashboard objects with additional features like graphs, alerts, etc.
Hiding system complexity and giving a clean interface (e.g., project management module)	Facade	Combine subsystems like TaskManager, Calendar, and Notifier behind a single interface
Dynamically creating objects for users (Admin, Member, Guest)	Factory	Return specific user class based on role or credentials
Handling request pipeline (e.g., request logging, validation, authentication)	Middleware / Chain of Responsibility	Each middleware processes the request and passes it to the next handler
Broadcasting updates (e.g., task assigned, message sent)	Observer	Notify all observers (e.g., team members) when a change occurs
Duplicating object configurations (e.g., template settings, profiles)	Prototype	Clone an existing object with the same settings instead of rebuilding from scratch
Restricting access to sensitive data (e.g., only Admins can view financials)	Proxy	Use a proxy to control access to sensitive operations based on role
Application configuration or logging instance	Singleton	One instance of configuration or logger throughout the project
Switching between strategies at runtime (e.g., sorting by date, priority, or status)	Strategy	Define interchangeable sorting or filtering strategies and switch them as needed

3.2 Implementation Using OOP Principles

Step	What to Do	Why It Matters
Apply OOP Principles	Use these basic concepts: • **Classes & Objects** – to represent things like User, Project, Task • **Inheritance** – for code reuse (e.g., Admin and Member both inherit from User) • **Encapsulation** – keep internal logic hidden • **Polymorphism** – same method, different behavior (Use of method overloading and overriding)	Makes your code clean, indented, reusable, and easy to maintain.

You Must include:

- At least 5 interacting classes with OOP concepts
- Use of at least 5 design patterns
- In report, you must explain the following:

OOP Explanation:

- -Why you defined each class
- -Where inheritance is used
- -How encapsulation is applied
- -Where polymorphism appears

Design Pattern Explanation:

- Which 5 patterns were used
- Where they are used in your code
- Why each pattern fits your problem

Task 4: Team Collaboration via GitHub

- Create a GitHub repository for the project
- Share your project to your team members
- Use branches for feature development
- Use pulls requests and code reviews for other members of your project
- Maintain a **README.md** with setup instructions
- We will check all commits which are initiated by different team members.

Task 5: API Testing using Postman

- Test the endpoint of your backends functionality using **Postman**
- Screenshots of Postman test cases or **Swagger documentation**

Example Endpoints to Test (It must be based on your project):

- POST /login
- GET /users
- POST /notifications
- PUT /settings/{id}
- DELETE /account/{id}
- And Others functionality

Task 6: Final Presentation & Report

- Live demo of the product Including Figma Design (**CI/CD preferred**)
- Submit a **final report** with the template provided.
- Project GitHub Link must be included in the report.

Mark Distribution:

	Marks
• Requirement Analysis	5
• UI/UX Design Using Figma	5
• Design Pattern and OOP Principles	5
• Team Collaboration via GitHub	10
• API Testing using Postman	5
• Final Presentation & Report	20
	Total Marks: 50

Assessment Criteria:

- Realistic and clear requirement analysis
- Working prototyping using Figma
- Successful implementation of OOP principles
- Functional team collaboration and clear backend functionality testing
- Problem-solving skills and ability to go beyond basic requirements