# Information regarding project implementation

## I. Descriptive Part

This section should briefly outline the importance of modelling in software engineering. Then, briefly describe the role of UML, with particular emphasis on the diagrams to be used in the practical section. Materials posted on https://www.uml-diagrams.org/ and other available resources on similar topics, including lecture slides, may be used.

## II. Practical Part

1) Study the examples of UML in system modelling posted at https://www.uml-diagrams.org/index-examples.html . Choose one example, analyse it in detail, and create and describe two diagrams that comprise the system model. Use one of the free online tools supporting UML (for example, Visual Paradigm https://www.visual-paradigm.com/ ).

2) Choose any one of the practical tasks proposed in Chapter IV and create a simple model of a hypothetical system in UML notation (several diagrams with descriptions).

## III. Organizational Information

The project should be completed systematically and submitted by the agreed-upon deadline in the form of a short report (8 to 10 pages). Documentation of work should be ongoing, so that the current status can be presented in class. The final version of the report should be sent to the instructor's email address.

## IV. Practical Tasks

### 1. Task Manager App

#### Project Overview

A simple web-based system for users to manage daily tasks — creating, updating, and organizing them by priority or deadline.

#### Stakeholders

- **Primary user:** Individual managing personal or team tasks.
- **System admin:** (Optional) manages user accounts.

#### Functional Requirements

1. The system shall allow users to **create, edit, and delete tasks**.
2. The system shall allow marking tasks as **complete/incomplete**.
3. The system shall allow filtering tasks by **status, date, or priority**.
4. The system shall persist task data in a **local or remote database**.
5. The system shall support **user login and logout** (optional).

#### Non-Functional Requirements

- **Usability:** Intuitive UI with minimal clicks to manage tasks.
- **Performance:** CRUD operations completed within 1 second.
- **Reliability:** No data loss after user logout or browser refresh.
- **Scalability:** Capable of handling 1000+ users concurrently (future).

### Modelling

- **Use Case Diagram:** Add Task, Edit Task, Delete Task, Mark Complete, View Tasks.
- **Activity Diagram:** Workflow from task creation → completion.
- **Entity Relationship Diagram (ERD):** Simple conceptual database model, e.g. User (1) — (M) Task.

## 2. Expense Tracker

### Project Overview

An application to log daily income and expenses, categorize transactions, and visualize spending.

### Stakeholders

- **User:** Person tracking finances.
- **System administrator:** Maintains user database and storage.

### Functional Requirements

1. The system shall allow users to **add, view, edit, and delete transactions**.
2. The system shall categorize expenses by **type (food, transport, etc.)**.
3. The system shall **generate reports** of monthly spending.
4. The system shall allow **exporting reports** (CSV/PDF).
5. The system shall display **graphs or charts** of expenses.

### Non-Functional Requirements

- **Security:** Financial data stored securely and encrypted.
- **Performance:** Data visualizations load under 3 seconds.
- **Compatibility:** Works on mobile and desktop browsers.

### Modelling

- **Use Case Diagram:** Manage Transactions, View Reports, Export Data.
- **DFD Level 0:** User → Expense Tracker → Database → Report Output.
- **ERD:** User, Transaction, Category (1-M, M-1).

## 3. Chat Application

### Project Overview

A real-time chat platform that allows users to communicate individually or in groups.

### Stakeholders

- **End user:** Sends and receives messages.
- **System admin:** Monitors users and chat logs.

### Functional Requirements

1. The system shall allow **user registration and login**.
2. The system shall support **real-time message transmission**.
3. The system shall allow **private and group chats**.
4. The system shall show **online/offline status**.
5. The system shall store **message history** for retrieval.

**Non-Functional Requirements**

- **Performance:** Messages delivered with latency < 1s.
- **Reliability:** Connection maintained even with intermittent internet.
- **Security:** Use HTTPS and encrypted sockets.
- **Scalability:** Support 100 concurrent users in prototype.

**Modelling**

- **Use Case Diagram:** Register, Login, Send Message, View Chat History.
- **Sequence Diagram:** Message sending between User A, Server, User B.
- **ERD:** User, ChatRoom, Message (M-M via ChatRoom).

## 4. Weather Forecast App

**Project Overview**

Displays current and forecast weather data for a given location using an external API.

**Stakeholders**

- **User:** Anyone who wants to check weather.
- **Weather data provider:** External API (e.g., OpenWeather).

**Functional Requirements**

1. The system shall allow the user to **search for a city**.
2. The system shall **fetch weather data** via an external API.
3. The system shall display **current and 5-day forecast**.
4. The system shall **cache recent searches** locally.
5. The system shall display **visual weather indicators**.

**Non-Functional Requirements**

- **Reliability:** Must handle API errors gracefully.
- **Usability:** Clean, responsive interface.
- **Availability:** App should function 99% uptime (when API available).
- **Maintainability:** API key stored securely in config.

**Modelling**

- **Use Case Diagram:** Search City, View Forecast, Save City.
- **DFD Level 1:** User → App → Weather API → Display Results.
- **ERD:** City, WeatherData, User (optional).

## 5. Library Management System

**Project Overview**

System for managing book lending, inventory, and borrower records.

**Stakeholders**

- **Librarian:** Manages books and transactions.
- **Member:** Borrows and returns books.
- **System admin:** Maintains users and backups.

**Functional Requirements**

1. The system shall allow **adding, editing, and removing books**.
2. The system shall allow **registering borrowers**.
3. The system shall record **book lending and return transactions**.
4. The system shall display **book availability status**.
5. The system shall generate **reports of overdue books**.

**Non-Functional Requirements**

- **Security:** Restricted access for librarian functions.
- **Reliability:** Accurate tracking of inventory.
- **Usability:** Intuitive UI for searching and issuing books.
- **Data Integrity:** No duplicate book IDs.

**Modelling**

- **Use Case Diagram:** Manage Books, Manage Members, Issue Book, Return Book.
- **Class Diagram:** Book, Member, Transaction.
- **ERD:** Member (1-M) Transaction (M-1) Book