**SDLC Case Study Worksheet**

Project Title: MedMate – Smart Prescription and Medicine Tracker\_\_

Team Name: **Team Orbit**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Team Members and Roles: **Muhammad Hasan Khan – Project Manager**\_\_\_

**Rayyan Ibrahim – UI/UX Designer\_\_\_\_\_\_ \_\_**

**Muhammad Mujtaba –Developer\_\_\_\_\_\_\_\_\_ \_\_**

**Husain Ahmad Khan – QA Tester\_\_\_\_\_\_\_\_\_\_\_\_**

**Muhammad Hamza Abbas – Deployment\_\_\_\_\_**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# **1. Requirements Phase**

Functional Requirements:

1. User Registration & Authentication – Users can sign up, log in, and log out securely.\_\_

2. Role Management – The system supports roles: User (patient/caregiver) and Admin.

3. Profile Management – Users can update profile details\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Prescription Upload – Users can upload prescriptions in image or digital format.\_\_\_\_\_\_

5. Prescription Management – Users can view, update, and delete prescriptions. \_\_\_\_\_\_

6. Reminder Scheduling – Users can set multiple reminders for each medicine.

7. Notification System – The app sends reminders via email

8. Mark as Taken – Users can mark medication as taken

9. Family Profile Support – A user can manage multiple family member’s prescriptions from one account.

10. Admin Dashboard – Admin can view/manage user accounts and monitor prescription logs.

11. Secure Authentication (JWT) – Backend uses JWT for safe user login sessions.

12. REST API Support – APIs for CRUD operations

13. Database Management – Store users, prescriptions, reminders, and family details securely.

14. Data Validation & Indexing – Database ensures integrity, with backup strategy support.

15. Frontend Features – Responsive UI with Login, Dashboard, Profile, Prescription & Reminder pages.

16. Form Validation & Error Handling – Prevents incorrect input during signup or prescription upload.

17. Deployment – Frontend on Vercel/Netlify, backend on Heroku/Render/Vercel, DB on MongoDB Atlas/PostgreSQL.

18. Documentation & Handover – User guides, technical docs, and source code delivered.

Non-Functional Requirements:

1. Performance – The system must support at least 100 concurrent users, with response times < 2 seconds.

2. Security – Strong encryption (HTTPS, JWT, password hashing, DB encryption).\_\_\_\_\_\_\_\_\_\_\_

3. Usability – Responsive and user-friendly interface across desktop, tablet, and mobile devices.

4. Scalability – The system must allow easy scaling for additional users, family members, and prescriptions.

5. Reliability/Availability – Maintain at least 99.5% uptime with reliable cloud deployment.

6. Maintainability – Modular and well-documented code for future updates.

7. Portability – Accessible on all major browsers (Chrome, Firefox, Edge, Safari).

# **2. Design Phase**

Draw a simple WBS (3 levels) and one UML diagram (use case/class diagram).  
Work Breakdown Structure (WBS): (Write as list or sketch tree diagram)

|  |
| --- |
| **1. Project Planning & Research** 1.1 Requirement Gathering 1.2 Stakeholder Identification 1.3 Scope Definition 1.4 Technical Feasibility Study 1.5 Project Timeline & Milestones  **2. UI/UX Design** 2.1 Wireframing (Login, Dashboard, Prescription, Reminders) 2.2 User Journey Mapping 2.3 Responsive Layouts 2.4 High-Fidelity UI Design (Figma) 2.5 User Feedback Collection  **3. Frontend Development (React.js)** 3.1 Project Setup & Routing 3.2 Component Development (Login, Signup, Dashboard, Prescription Upload, Reminder View) 3.3 State Management (Redux/Context API) 3.4 Form Validation & Error Handling 3.5 Integration with Backend APIs 3.6 Testing & Responsiveness  **4. Backend Development (Node.js + Express.js)** 4.1 API Structure & Initialization 4.2 Authentication & Authorization (JWT) 4.3 REST APIs (Users, Prescriptions, Reminders, Family Profiles) 4.4 Role Management (User/Admin) 4.5 Middleware & Error Handling 4.6 API Testing (Postman)  **5. Database (MongoDB/PostgreSQL)** 5.1 Schema Design (Users, Prescriptions, Reminders, Family Profiles) 5.2 CRUD Operations 5.3 Data Validation & Indexing 5.4 Backup Strategy  **6. Admin Panel** 6.1 User Management 6.2 Prescription Monitoring  **7. Testing & Debugging** 7.1 Unit Testing 7.2 Integration Testing 7.3 UI Testing (Responsive, Mobile View) 7.4 Bug Fixing & QA Review  **8. Deployment & Handover** 8.1 Final Build & Optimization 8.2 Frontend Deployment (Vercel/Netlify) 8.3 Backend Deployment (Heroku/Render) 8.4 Database Setup (MongoDB Atlas/Postgres) 8.5 Documentation & Handover |

UML Sketch (Use Case / Class Diagram/sequence Diagram): (Draw below)

|  |
| --- |
|  |

# **3. Backend Design**

Attach you design file. And mention the tool that you used.

**4. Development Phase**

Write pseudo-code/code for one key function in your system.  
Pseudo-code Example:  
  
FUNCTION scheduleReminder(prescriptionID, time, userID)

INPUT: prescriptionID, time, userID

PROCESS:

1. Validate that prescriptionID exists in database.

2. Check userID permissions.

3. Save reminder details (prescriptionID, userID, time, status="pending").

4. Schedule notification service at given time.

OUTPUT: Confirmation that reminder has been scheduled successfully.

END FUNCTION

# **5. Testing Phase**

Write 3 test cases.

|  |  |
| --- | --- |
| **Test Case ID** | **Description** |
| TC-01 | Verify user login with correct credentials. |
| TC-02 | Verify prescription upload. |
| TC-03 | Verify reminder notification. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Input(s)** | **Expected Output** | **Result (Pass/Fail)** |
| TC-01 | Email = test@medmate.com, Password = 123456 | Redirect to Dashboard. | Pass |
| TC-02 | Upload prescription image | Prescription stored in DB, success message shown. | Pass |
| TC-03 | Reminder set for 9:00 AM. | User receives notification at 9:00 AM. | Pass |

# **6. Reflection**

1. Which SDLC phase was the most challenging? Why?  
 The **Design Phase** was the most challenging because it required creating WBS, UML diagrams, and database schema that aligned with all functional requirements.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Which SDLC model (Waterfall, Agile) best fits this project? Why?  
 **Agile** fits best because the project has multiple modules (User, Prescription, Reminders, Admin Panel) and can be developed iteratively with weekly sprints. It also allows flexibility for changes during development.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. How you determine functional and non-functional requirements?  
 Functional requirements were derived directly from **project scope and objectives** (what the system must do).

Non-functional requirements were derived from **performance, security, usability, and deployment needs** (how the system should behave).

# **7. Attachments**

Along with this filled worksheet, provide supported document of design methodologies/diagrams and document of types of testing techniques.