Event Countdown App

DEPI Final Project
3/21/25

Team Members:

- **Ahmed Abdelmajeed Abdelmonsef**
- **Ahmed Salem Hussein**
- **Dina Abdullah Hussein**
- **Maria Yousef Makram**
- **Ibrahim Ahmed Ibrahim**

Table of Contents:

Event Countdown

1. Introduction

- **Overview of the Project**
  The **Event Countdown App** is a mobile application designed to help users efficiently track and manage upcoming events. It offers features such as event creation, countdown timers, notifications, and organizational tools to ensure users never miss an important date.

- **Problem Statement**
  Many individuals struggle to keep track of important events, leading to missed deadlines and disorganization. This app provides a structured solution by allowing users to manage events through an intuitive interface and timely reminders.

- **Objectives**
  - Provide an easy-to-use platform for managing events.
  - Allow users to set countdown timers for upcoming events.
  - Send timely notifications to ensure users stay informed.
  - Improve productivity and time management.

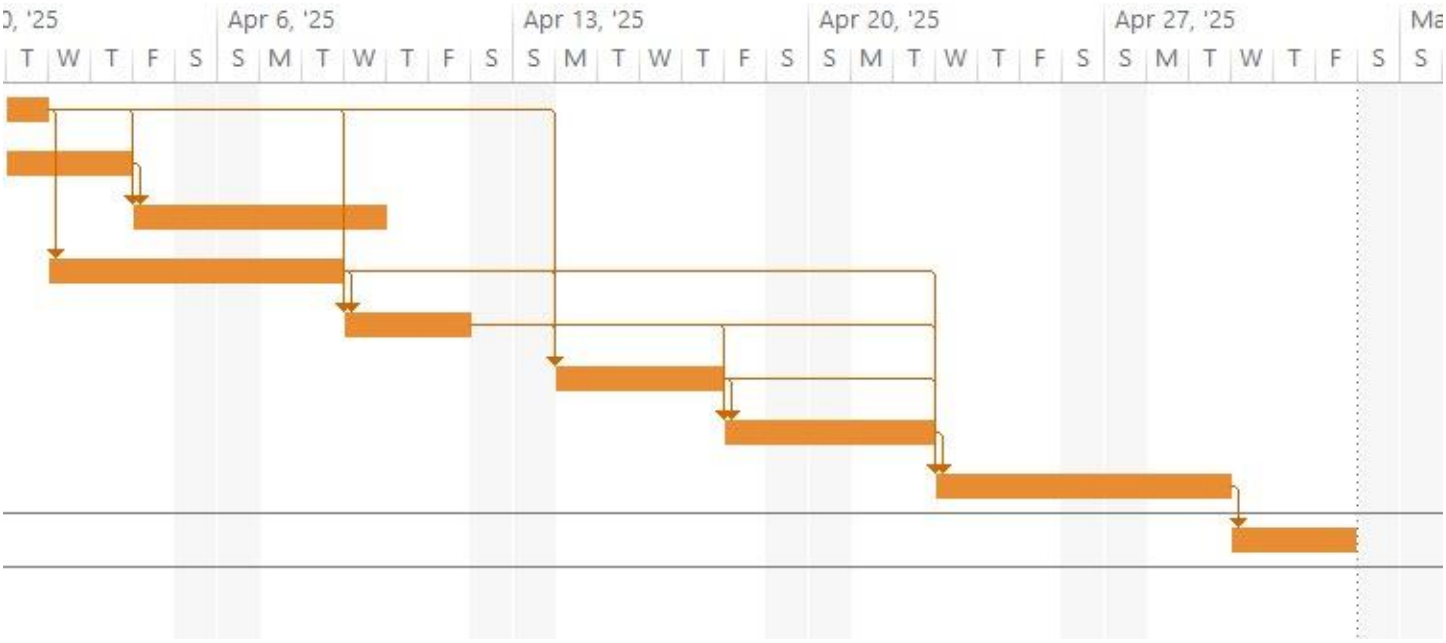
- **Target Audience**
  - Individuals seeking better personal organization.
  - Small businesses and teams managing deadlines and meetings.
  - Students tracking exam schedules and academic activities.

2. Project Planning & Management

- **Project Proposal**
  - A high-level overview of the app, including:
    a. **Objective:** Help users track and manage event countdowns with notifications.
    b. **Scope:**
       - Users can add, edit, and delete countdown events.
       - Push notifications remind users about upcoming events.
       - Events are stored locally or in the cloud.
    c. **Expected Outcome:** Android mobile app with an intuitive UI.

- **Project Plan**
  - **Gantt Chart:**

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		Setup Android & project	1 day	Thu 3/20/25	Thu 3/20/25	
2		UI Prototyping	3 days	Thu 3/20/25	Mon 3/24/25	
3		Basic UI Development	4 days	Tue 3/25/25	Fri 3/28/25	1,2
4		Event Management Features	5 days	Fri 3/21/25	Thu 3/27/25	1
5		Countdown Logic	3 days	Fri 3/28/25	Tue 4/1/25	1,4
6		Push Notifications	4 days	Wed 4/2/25	Mon 4/7/25	1,4,5
7		Custom Reminder Settings	3 days	Tue 4/8/25	Thu 4/10/25	5,6
8		Testing & Debugging	5 days	Fri 4/11/25	Thu 4/17/25	4,5,6,7
9		Deployment & Documentation	3 days	Fri 4/18/25	Tue 4/22/25	8
10						



- **Task Assignment & Roles**

1. **Project Manager:** Oversees the project timeline and deliverables.
2. **UI/UX Designer:** Designs wireframes and prototypes.
3. **Frontend Developer:** Implements the user interface.
4. **Backend Developer:** Handles database and API integration.
5. **Tester:** Conducts testing and bug reporting.

- **Risk Assessment & Mitigation**

RISK	Impact	Mitigation Strategy
Push notifications fail on some devices	High	Use Firebase Cloud Messaging (FCM) and test across different Android versions.
Countdown timer stops updating in background	High	Use WorkManager for background execution. Handle Doze Mode restrictions.
App crashes due to memory leaks	Medium	Use Android ViewModel & LiveData to manage lifecycle events. Test memory usage.
Low performance on older devices	Medium	Optimize UI rendering, minimize heavy animations, and use RecyclerView efficiently.
Data loss if user uninstalls the app	Medium	Offer cloud sync (Firebase Firestore) or backup options with Google Drive API.
Play Store rejection due to policy issues	High	Follow Google Play Store guidelines (permissions, security, data privacy).
Too many notifications annoy users	Medium	Allow users to customize notification frequency in app settings.

- **Key Performance Indicators (KPIs)**

KPI	Description	Target Goal
Countdown Accuracy	Ensure countdown timers update smoothly in foreground & background.	99% accuracy in time calculations.
Notification Reliability	Push notifications should trigger at the correct scheduled time.	Less than 2% failure rate.
Performance Optimization	The app should run efficiently without memory leaks or crashes.	Crash-free rate of > **95%**.
User Experience	Measure user satisfaction based on app reviews.	**4.5+ rating** on Google Play Store.
Response Time	Time taken to open an event and display countdown.	< **500ms** load time.
System Uptime	The app should function properly without unexpected crashes.	**99.9% uptime**.
User Adoption Rate	Number of new users installing and using the app.	**10,000+ downloads** in the first 6 months.

3. Literature Review & Research

Competitor Analysis

- **Competitors:** Apps like "Countdown Timer" and "EventTracker."
- **Strengths:** Simple UI, reliable notifications.
- **Weaknesses:** Limited customization and calendar integration.

Technology Research

- **Frontend:** Android JetPack Compose (A modern framework for building flexible and intuitive Android UIs)
- **Backend:** Firebase for real-time database and authentication.
- **Database:** Cloud Firestore to store user events and support real-time data synchronization.
- **Push Notifications:** Firebase Cloud Messaging (FCM).

4.Requirements Gathering

- ## Stakeholder Analysis

The system serves various stakeholders with unique requirements:

1. Users (Event Creators & Participants)

Create, modify, and delete events.

See countdowns for events to occur.

Receive event reminders and notifications.

2. System Administrators

Manage the system for seamless running.

Manage user accounts and tech support.

3. Notification Service

Manages reminders programmed ahead of time and event notices.

Notifies in a timely fashion using push notifications, emails, or SMS.

- ## User Stories & Use Cases

User stories define how users interact with the system:

1. **As a user, I want to create an event so that I can monitor upcoming events.**
2. **As a user, I want to update or remove my event in case I wish to alter something.**
3. **As a user, I want to have a countdown so that I will realize how much time left for my event.**
4. **As a user, I want to receive notifications so that I will remember a future event.**
5. **As an administrator, I want to manage system users for maintaining system integrity.**

- ## Use Case Example: Creating an Event

Use Case Name	Create Event
Actor(s)	User

Precondition	User must be logged in
Description	User inputs event details and submits the form
Postcondition	Event is stored in the database, and a notification is scheduled
Alternative Flow	If input is invalid, show an error message

- **Functional Requirements**

The system shall provide the following functionalities:

1. User Management

- **User registration and login.**
- **Profile management.**

2. Event Management

- **Create, edit, and delete events.**
- **View event countdown.**
- **Enable or disable notifications.**

3. Notification Service

- **Send reminders at a specified time to users.**
- **Handle multiple notification channels (push, email, SMS).**

4. Data Storage & Retrieval

- **Store event data in a database.**
- **Fetch event data efficiently to show.**

5. Non-Functional Requirements

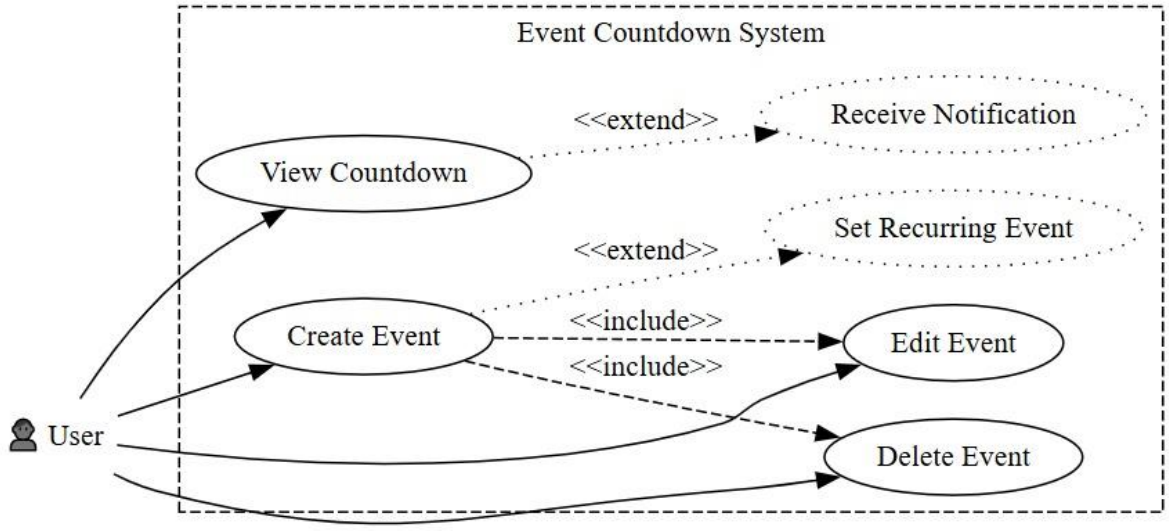
- **The system should meet the following quality requirements:**
- **Performance: Should be capable of handling multiple users creating and showing events without delay.**
- **Security: Secure and safe authentication and data encryption.**
- **Usability: Clean and simple UI for user friendliness.**
- **Reliability: Notifications should be sent properly and at the right time.**
- **Scalability: Capable of handling more users and events.**

---

5. System Analysis & Design

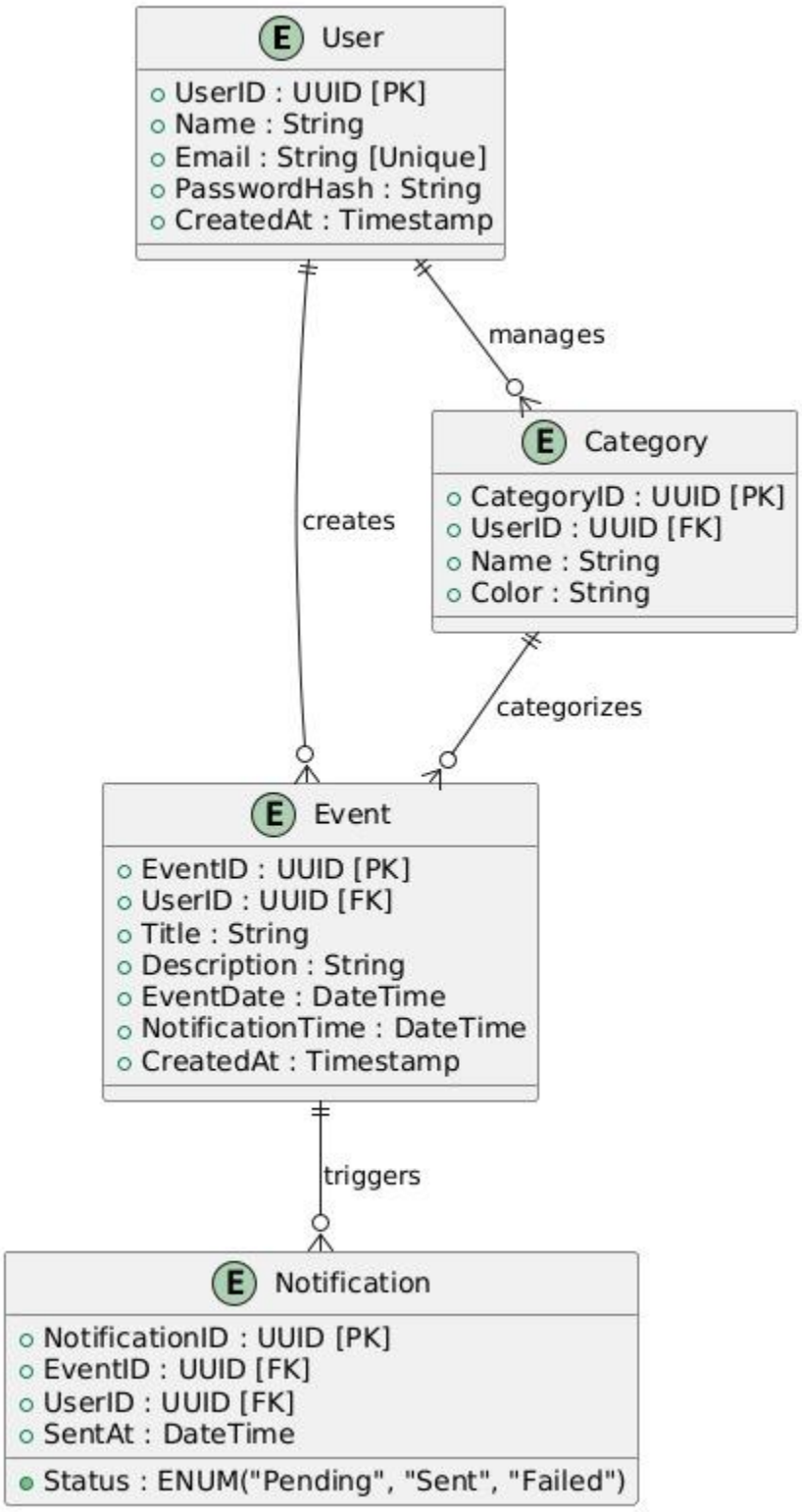
- **Problem Statement & Objectives**
- **Define the Problem:** The issue being solved and the goals of the system.

1. Use Case Diagram:



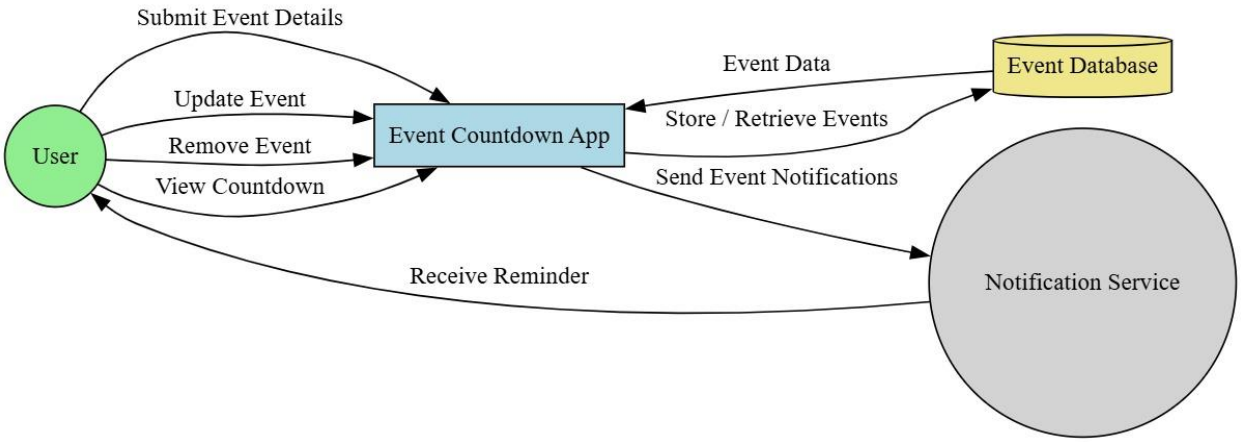
2. Database Design & Data Modeling

- **ER Diagram:** Entity-Relationship Diagram defining database structure.

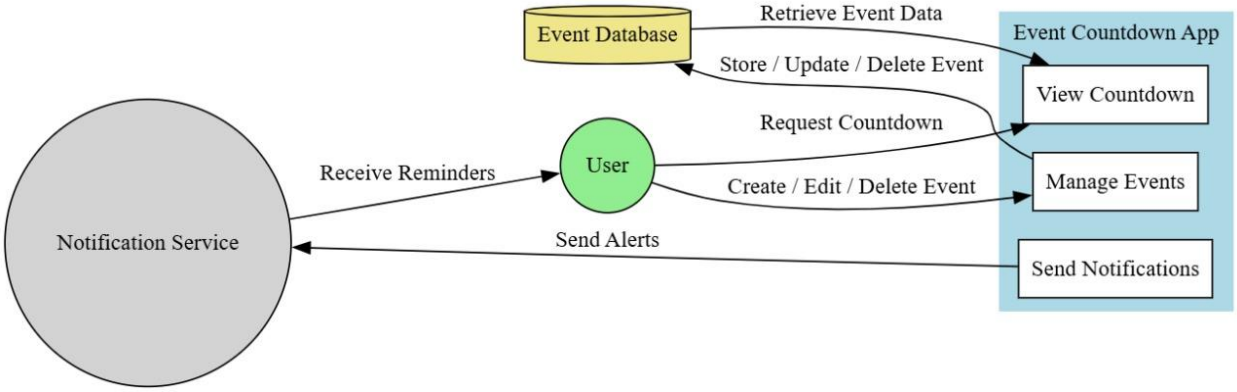


- **Data Flow & System Behavior**
- **DFD:** Data Flow Diagram (Context & Detailed Levels).

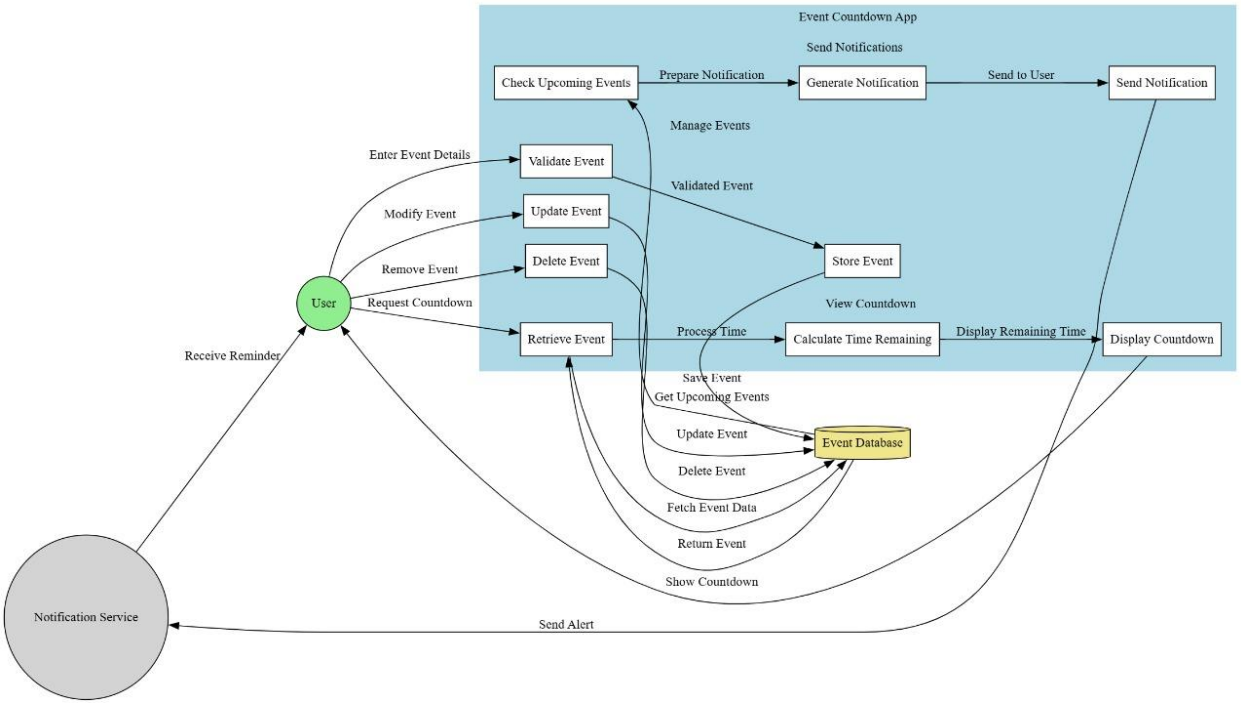
1. DFD Level 0 (Context Diagram)



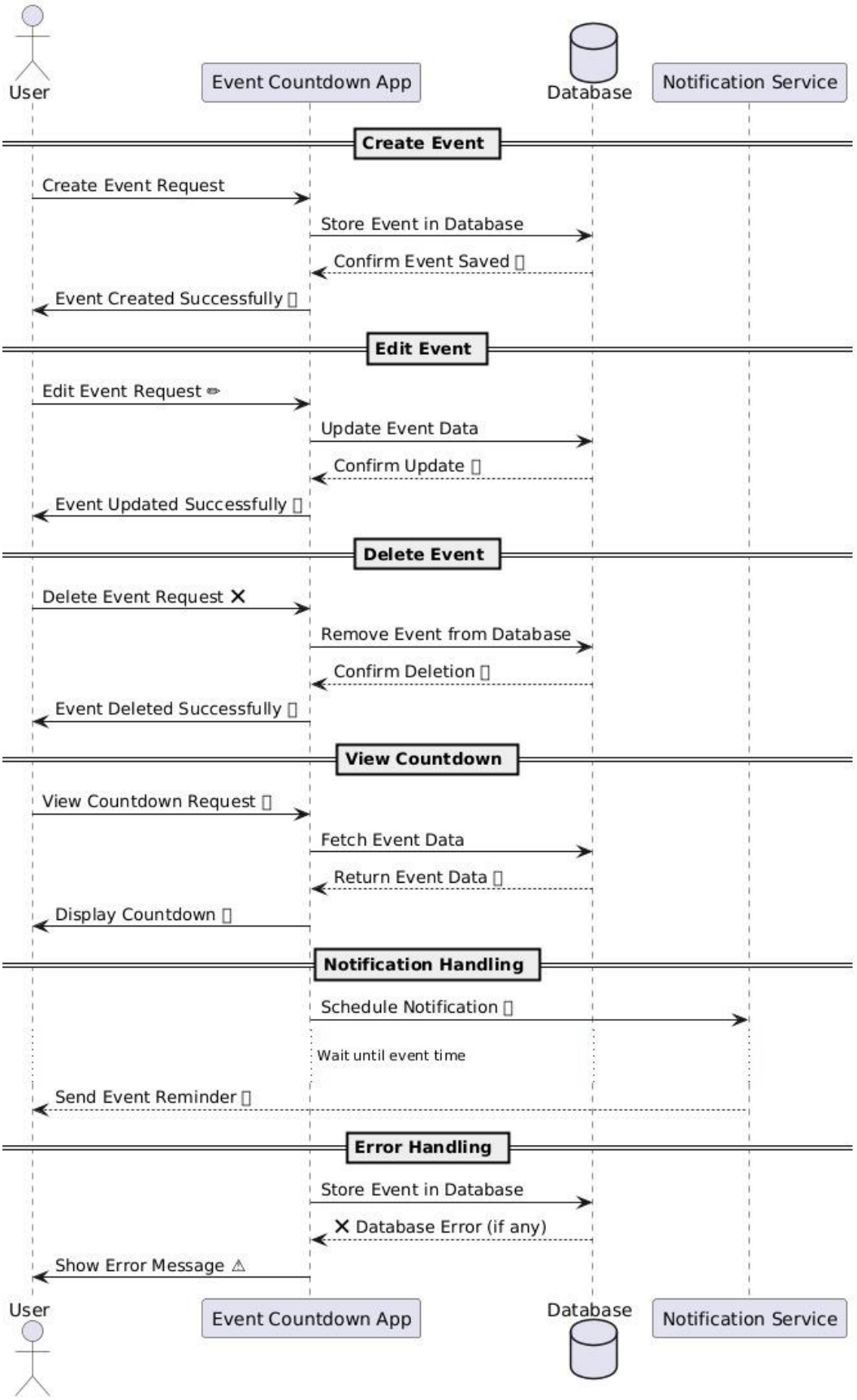
2. Data Flow Diagram (DFD) - Level 1



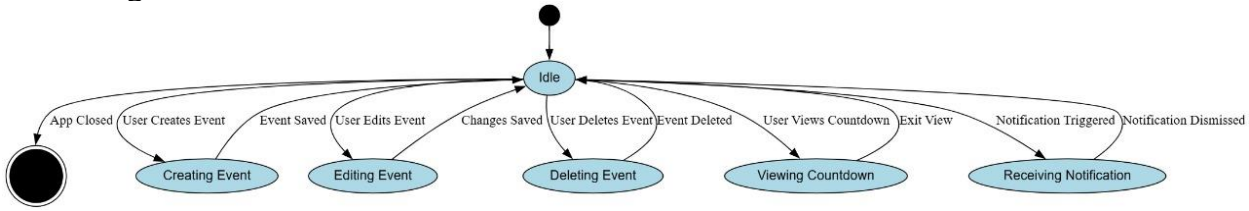
3. Data Flow Diagram (DFD) - Level 2



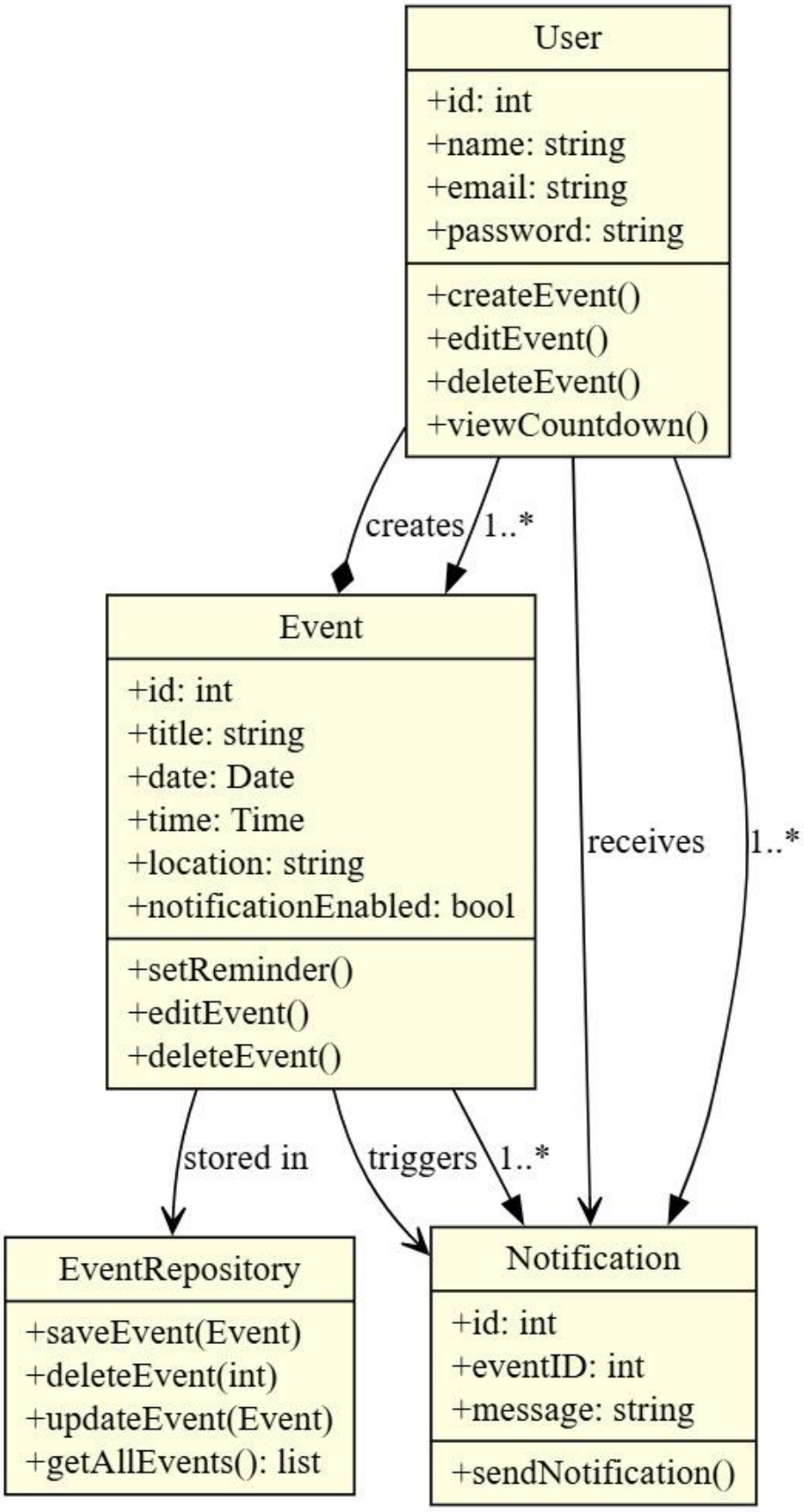
- **Sequence Diagrams:**


User
Event Countdown App
Database
Notification Service
Create Event
Create Event Request
Store Event in Database
Confirm Event Saved
Event Created Successfully
Edit Event
Edit Event Request
Update Event Data
Confirm Update
Event Updated Successfully
Delete Event
Delete Event Request
Remove Event from Database
Confirm Deletion
Event Deleted Successfully
View Countdown
View Countdown Request
Fetch Event Data
Return Event Data
Display Countdown
Notification Handling
Schedule Notification
Wait until event time
Send Event Reminder
Error Handling
Store Event in Database
Database Error (if any)
Show Error Message
User
Event Countdown App
Database
Notification Service

- **State Diagram**



- **Class Diagram:**


User
+id: int
+name: string
+email: string
+password: string
+createEvent()
+editEvent()
+deleteEvent()
+viewCountdown()
creates
1..*
Event
+id: int
+title: string
+date: Date
+time: Time
+location: string
+notificationEnabled: bool
+setReminder()
+editEvent()
+deleteEvent()
receives
1..*
stored in
triggers
1..*
EventRepository
+saveEvent(Event)
+deleteEvent(int)
+updateEvent(Event)
+getAllEvents(): list
Notification
+id: int
+eventID: int
+message: string
+sendNotification()

- # 5. Implementation (Source Code & Execution)
- **Source Code**
- **Code Structure:** Well-organized, commented code following best practices.
- **Coding Standards:** Consistent formatting and meaningful variable names.
- **Modular Design:** Reusable components, functions, and classes.
- **Security & Error Handling:** Secure coding practices and validation checks.
- **Version Control & Collaboration**
- **Repository:** Hosted on GitHub/GitLab with a link.
- **Branching Strategy:** Defined workflow (GitFlow, Feature Branching).
- **Commit History:** Meaningful commit messages and pull request descriptions.
- **CI/CD Integration:** Automated builds, testing, and deployment (if applicable).
- **Deployment & Execution**
- **README File:** Includes installation steps, system requirements, configuration, and execution guide. **Executable Files & Deployment Link:** Web/Mobile app access details.

- # 6. Testing & Quality Assurance
- **Test Cases & Test Plan**
- **Test Scenarios & Expected Outcomes:** Defined testing methodology.
- **Automated Testing:** Scripts for automated verification.
- **Bug Reports:** List of issues identified and resolved.

- # 7. Final Presentation & Reports
- **User Manual**
- **Instructions for End Users:** How to use the system effectively.
- **Technical Documentation**
- **System Architecture:** Overview of database schema and API documentation.
- **Project Presentation (PPT/PDF)**
- **Summary of Project:** Key challenges, solutions, and results.
- **Video Demonstration (Optional)**
- **Project Demo:** Short video showcasing functionalities.
