To-Do List Application Report

Title Page

Project Title:To-Do List Application

Team Members:

- Name 1, Roll No: Hashir Ali R056

- Name 2, Roll No: Maaz Ahmed R060

- Name 3, Roll No: Muhammad Huzaifa R053

- Name 4, Roll No: Muhammad Naveed R045

Table of Contents

To-Do List Application - Use Case Specification

1. **Introduction**

This document specifies a comprehensive use case specification for the To-Do List Application, detailing the functional requirements and interactions between users and the system.

2. Use Case Actors

2.1 Primary Actors

User: The primary actor who will interact with the To-Do List application

Administrator: Manages system-level configurations and user accounts (optional)

3. Use Case Diagram Overview

3.1 System Boundary

The application of To-Do List involves the following primary use cases:

Task Management

List Management User

Authentication

Notification Management

4. Extended Use Cases

4.1 Task Creation Use Case

Actor: User

Description: The system allows users to create new tasks

Preconditions:

The user is authenticated

The user has at least one accessible task list

Main Flow:

The user selects "Create New Task"

System displays task creation form

User enters task details (title, description, due date, priority)

System validates task information

System saves the task to the selected list

Alternative Flows:

User can cancel task creation

User can add subtasks to the main task

4.2 Task Management Use Case

Actor: User

Description: Comprehensive task management capabilities

Use Case Scenarios:

Edit Task

Change task details

Change task status

Change priority

Delete Task

Delete individual tasks

Delete multiple tasks

Mark Task Complete

Change task status to completed

Optional archiving of completed tasks

4.3 List Management Use Case

Actor: User

Description: Creates, manages, and organizes task lists

Functional Requirements:

Create new task lists

Rename existing lists

Delete lists

Share lists with other users

Set list-level permissions

4.4 User Authentication Use Case

Actor: User

Description: Secure access to the application for users

Authentication Flows

User Registration

New account

Email and password

Social media authentication is optional

User Login

Enter credentials

Authenticate against system records

Password Management

Forgotten password reset

Change existing password

4.5 Notification Management Use Case

Actor: User

Description: Manage task and system notifications

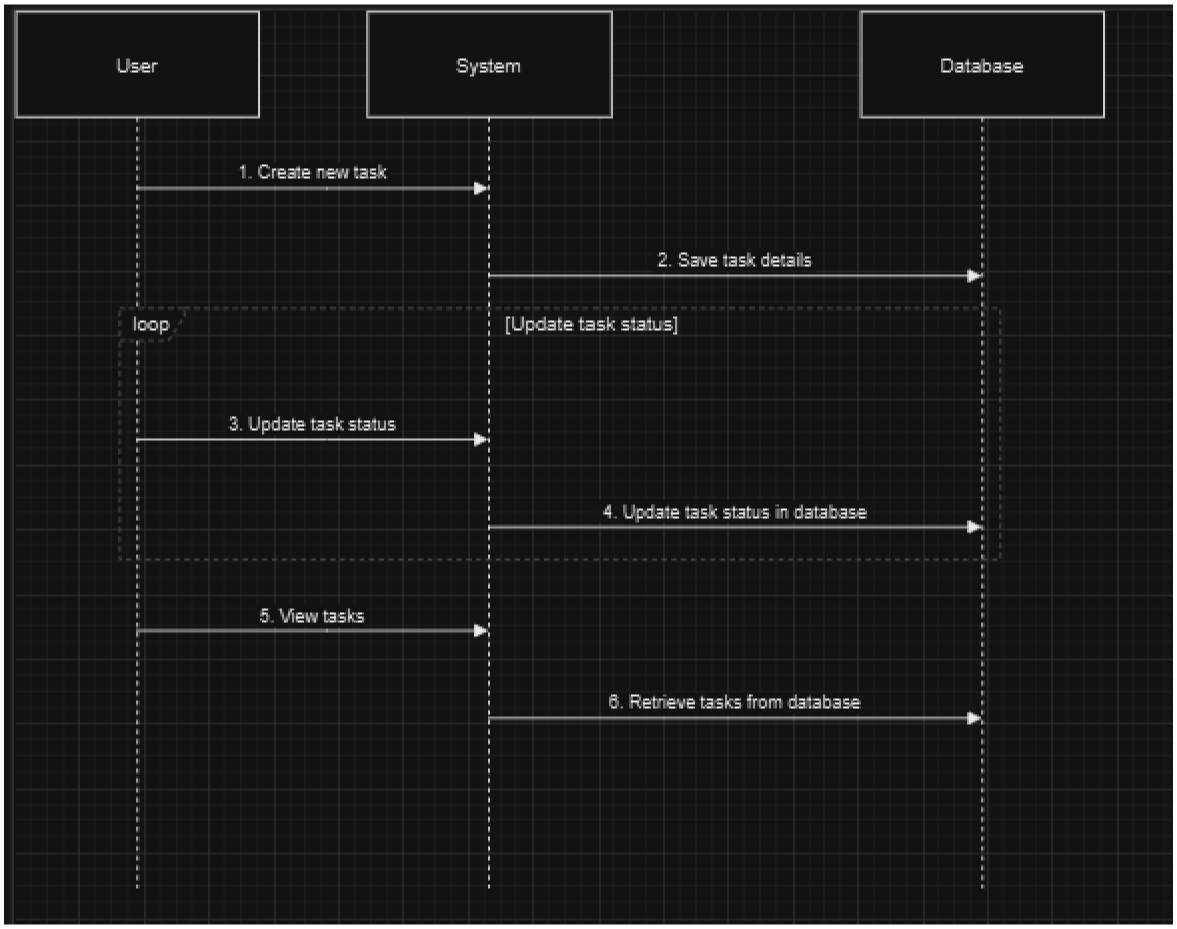
Notification Types:

Task Due Date Reminders

Overdue Task Alerts

Shared List Updates

System Ann



To-Do List Application - Activity Diagram Specification

1. Introduction

1.1 Purpose

This document describes the activity diagram specification for the To-Do List Application, outlining the workflow and process flows of key system functionalities.

2. Activity Diagram Overview

2.1 Scope

The activity diagram represents the dynamic behavior and workflow of critical processes in the To-Do List application, including:

User Authentication

Task Creation

Task Management

List Management

3. Activity Flows with Detailed Information

3.1 User Authentication Activity Flow

3.1.1 Registration Process

```
Start → Enter Registration Details

↓

Validate User Information

  ├─ Invalid Information → Display Error Message

  └─ Valid Information → Create User Account

↓

Send Verification Email

↓

Confirm Email Verification

↓

End (User Registered)
```

3.1.2 Login Process

```
Start → Navigate to Login Page

↓

Enter Credentials

↓

Authenticate Credentials

  ├─ Authentication Fails → Display Error Message

  └─ Authentication Succeeds → Access User Dashboard

↓

Load User's Task Lists and Pending Tasks

↓

End (User Logged In)
```

3.2 Task Creation Activity Flow

3.2.1 Create New Task

Start → Select "Create Task"

↓

Select Task List

↓

Input Task Details

- Title
- Description
- Due Date
- Priority

↓

Validate Task Information

```
 ├── Invalid Information → Highlight Errors
 └── Valid Information → Save Task
```

↓

Update Task List

↓

Send Optional Notification

↓

End (Task Created)

3.3 Task Management Activity Flow

3.3.1 Update Task Status

Start → Select Existing Task

↓

Choose Action

 Mark as Complete

  Update Task Status

Move to Completed Tasks

Edit Task Details

Modify Task Attributes

Save Changes

Delete Task

Confirm Deletion

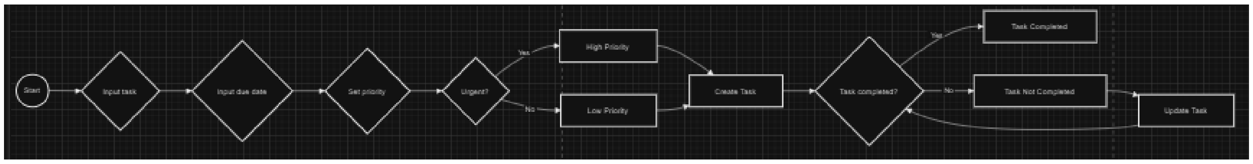
Remove Task from List

↓

Sync Changes

↓

End (Task Updated)



To-Do List Application - Class Diagram Specification

1. Introduction

1.1 Purpose

This document is a class diagram specification for the To-Do List Application, describing the system's structural design, class relationships, and major object interactions.

2. Class Overview

2.1 Core Classes

The core classes include:

User Task

TaskList

Notification

Authentication

Permission

Relationship Descriptions

A User can own multiple TaskLists

A User can be assigned multiple Tasks A

TaskList contains multiple Tasks

A Task can have multiple Subtasks

A User receives multiple Notifications

3. Design Patterns

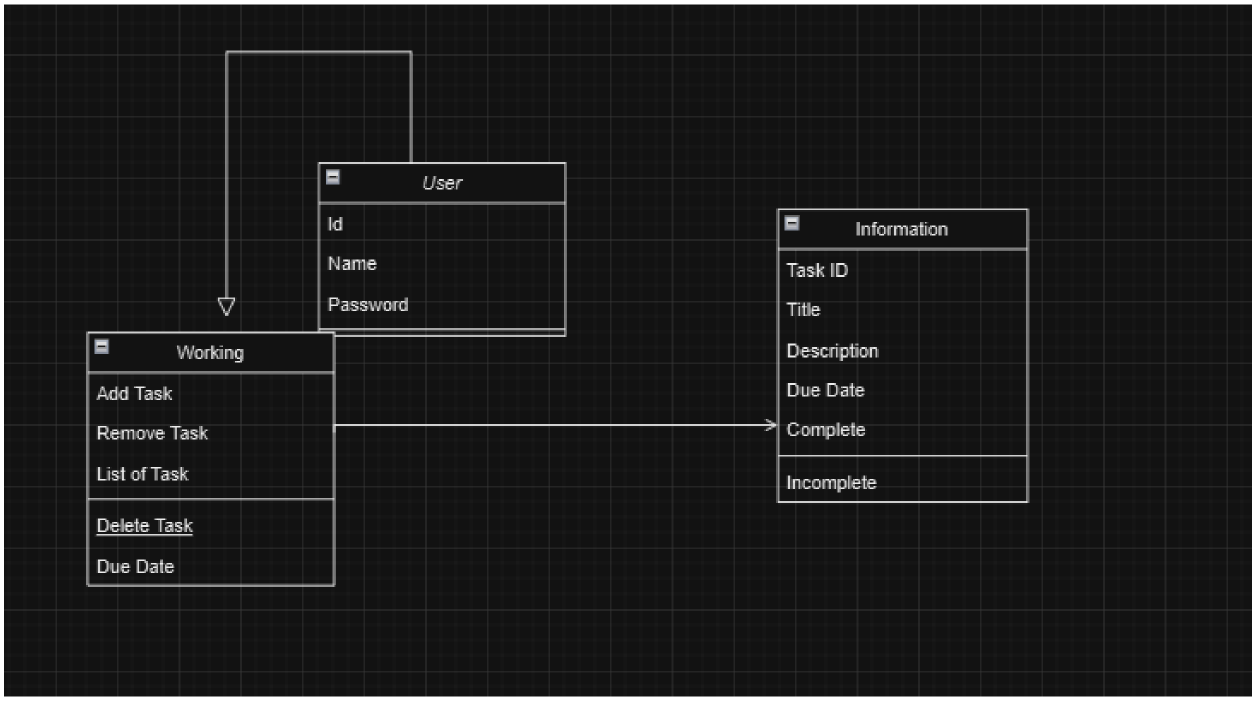
3.1 Applied Patterns

Singleton: Authentication Service

Factory: Task and Notification Creation

Observer: Notification System

Strategy: Permission Management



To-Do List Application - State Diagram Specification

Purpose
The state diagram represents the workflow of a task in the To-Do List Application. It shows the transitions between various task states and the events that trigger these transitions.

States and Transitions

1. **To_Do_List (Initial State)**
   - This is the starting state of the workflow where tasks are listed but not yet started.
   - **Transition**:
     - **Start Task**: Initiates a task and transitions the state to **In_Progress**.
2. **In_Progress**
   - This state indicates that the task is currently being worked on.
   - **Transitions**:
     - **Complete Task**: When the task is finished successfully, it transitions to the **Completed** state.
     - **Task not completed on time**: If the task exceeds its deadline, it moves to the **Overdue** state.
3. **Completed**
   - This state signifies that the task has been successfully finished.
   - There are no further transitions from this state.
4. **Overdue**
   - This state represents tasks that were not completed within the specified time.

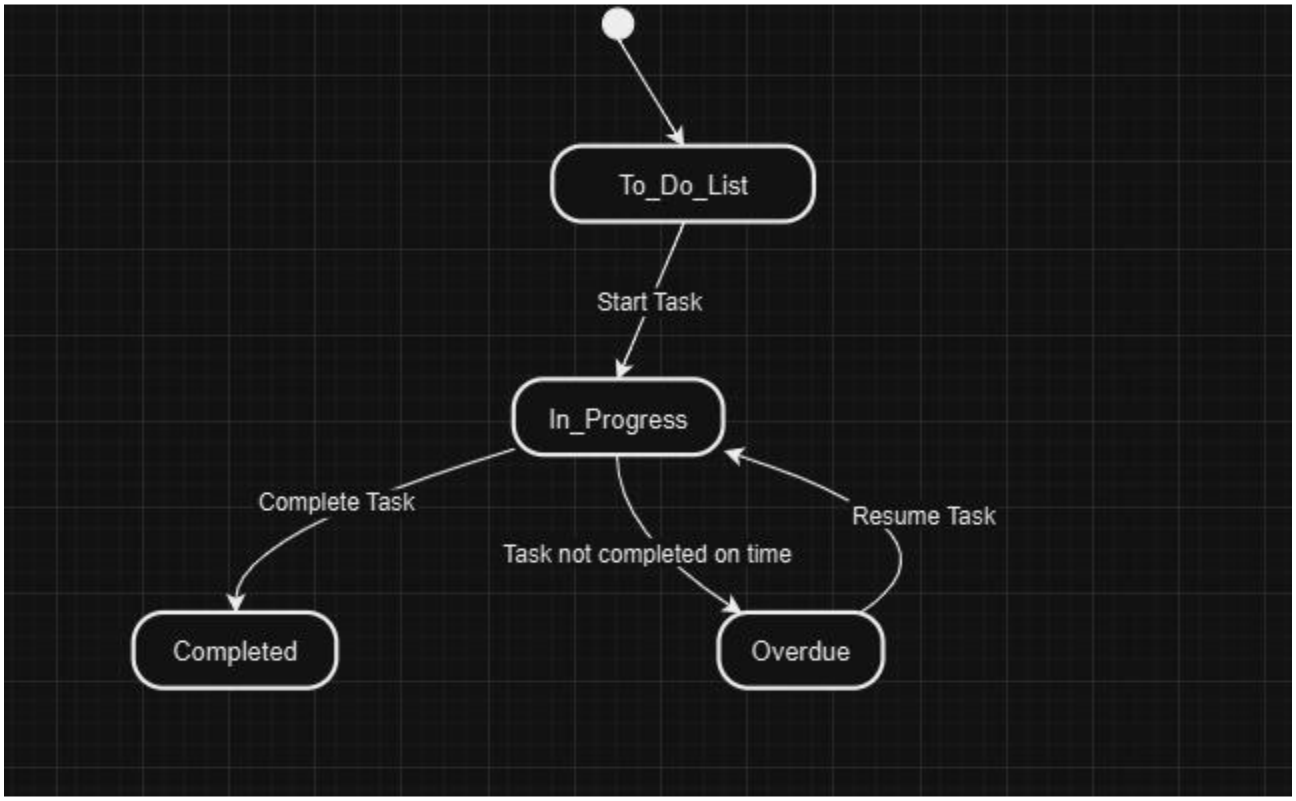
- **Transition**:
    - **Resume Task**: Allows the task to be resumed and moves it back to the **In_Progress** state.

Workflow Summary

- The task starts in the **To_Do_List** state.
- It progresses to **In_Progress** when work begins.
- From **In_Progress**, the task can either move to **Completed** if successfully finished or to **Overdue** if not completed on time.
- Overdue tasks can be resumed, transitioning them back to **In_Progress**.

Key Notes

- The diagram clearly defines states and transitions to ensure tasks follow a structured workflow.
- Arrows indicate the flow of transitions triggered by specific events (e.g., *Start Task*, *Complete Task*, *Task not completed on time*, and *Resume Task*).



To-DoListApplication - Sequence Diagram Specification

1. Introduction

1.1 Purpose

This document provides an in-depth specification of a sequence diagram that illustrates dynamic interactions among components of the To-Do List Application under key user scenarios.

2. Sequence Diagram Overview

2.1 Covered Scenarios

Registration and authentication of users

Task creation

Task update and change of status

Sharing lists

Generation of notifications

3. Detailed Sequence Flows

3.1 User Registration Sequence

Participant: User Interface

Participant: Authentication Service

Participant: User Repository

Participant: Notification Service

Process:

User inputs registration information

Client-side validation is done by UI

Registration request is forwarded to Authentication Service

Authentication Service validates details

User account is created in User Repository

Verification token is generated

Verification email is sent through Notification Service

Return registration success

Sequence Steps:

User Interface requests registration

Client-side input validation is performed

Request for registration is sent to Authentication Service

Server-side comprehensive validation

User account creation in database

Generation of verification token

Sending of email notification

Return confirmation of registration

3.2 Task Creation Sequence

Participant: User Interface

Participant: Task Service

Participant: Authorization Service

Participant: Task Repository

Participant: Notification Service

Flow:

User requests task creation UI

prepares task details Send

task creation request

Authorization Service validates user permissions

Task Service processes task

Persist task in Task Repository

Generate task creation notification Return

task creation confirmation **Sequence**

Steps:

Validate permission before creating the task

Atomic creation and persistence of the task

Auto-notification generation

Instant feedback to the user

3.3 Task Update Sequence

Participant: User Interface

Participant: Task Service

Participant: Authorization Service

Participant: Task Repository

Participant: Notification Service

Flow:

The user chooses a task to update

Modify task information

Make update request

The Authorization Service verifies update permission

Task Service verifies changes for acceptance

Update the task in the Task Repository

Create an update notification

Broadcast update to list owners

3.4 List Sharing Sequence

Participant: User Interface

Participant: List Service

Participant: Authorization Service

Participant: User Repository

Participant: Notification Service

Flow:

User requests list sharing

Input sharing information (email/username)

Verify recipient

Check sharing permissions

Create sharing invitation

Send invitation notification

Update list permissions

Notify list owner

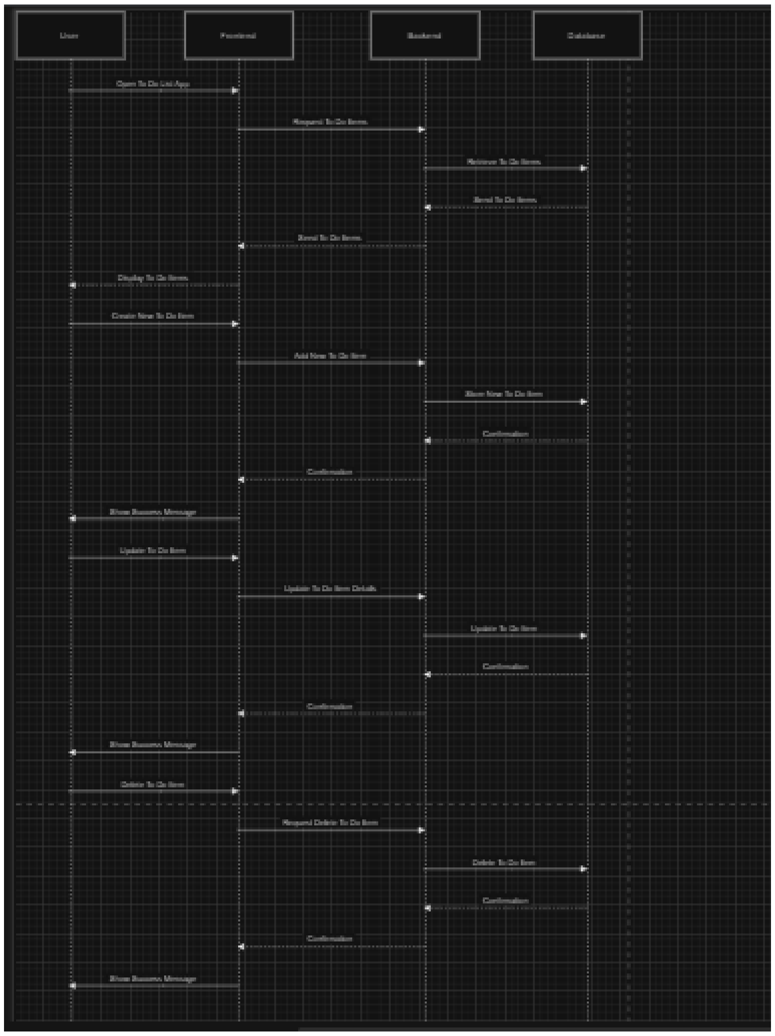
Sequence Steps:

Recipient validation

Permission-based sharing

Invitation mechanism

Collaborative update notifications



To-Do List Application - Collabration Diagram Specification

1. Purpose

The collaboration diagram illustrates the interactions between objects in the To-Do List Application during task creation. It highlights the flow of messages and the relationships between various components**.**

Collaboration Diagram for Task Creation

Objects Involved:

- **User**: The actor initiating the process.
- **UI**: User interface facilitating the interaction.
- **AuthService**: Handles permission validation.
- **TaskService**: Processes the task request.

- **TaskRepository**: Responsible for saving task data.
- **NotificationService**: Sends a confirmation message.

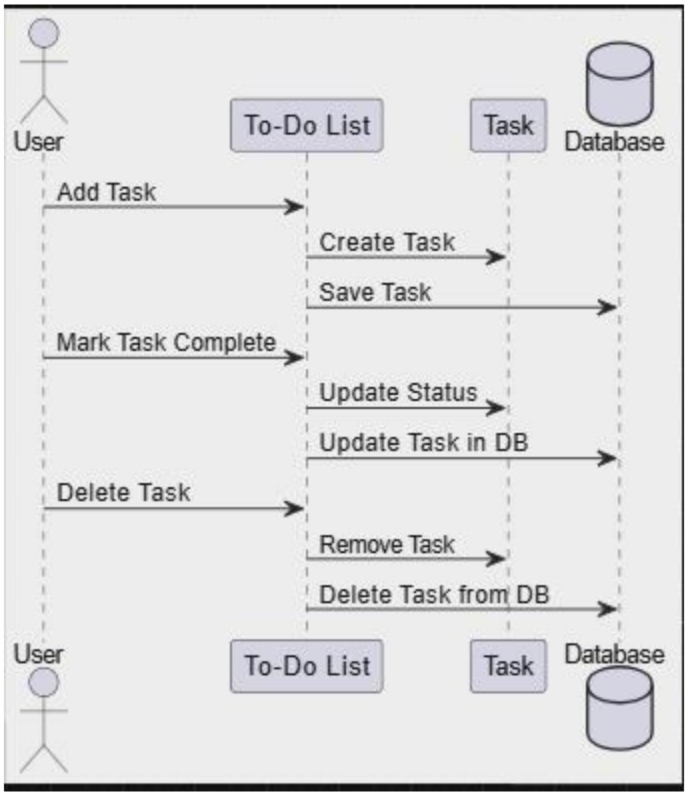
Message Flow:

1. **User → UI**: Initiates the `Create Task` action.
2. **UI → AuthService**: Requests permission validation for the task creation.
3. **AuthService → TaskService**: Sends validation confirmation and forwards the task creation request.
4. **TaskService → TaskRepository**: Requests task data to be saved in the database.
5. **TaskRepository → TaskService**: Confirms the task has been successfully saved.
6. **TaskService → NotificationService**: Generates a confirmation notification.
7. **NotificationService → User**: Sends the confirmation notification to the user.

Visual Representation
The collaboration diagram is represented by showing objects as rectangles and messages as labeled arrows connecting the objects in the order they interact.

- Place **User**, **UI**, **AuthService**, **TaskService**, **TaskRepository**, and **NotificationService** as objects.
- Use numbered arrows to denote the sequence of interactions.

The sequence of interactions would resemble the following:
1 → User interacts with UI.
2 → UI communicates with AuthService.
3 → AuthService confirms permissions to TaskService.
4 → TaskService saves the task in TaskRepository.
5 → TaskRepository confirms back to TaskService.
6 → TaskService initiates notification generation in NotificationService.
7 → NotificationService sends confirmation to User.



To-Do List Application - Data Flow Diagram (DFD) Specification

1. Introduction

1.1 Purpose

This document contains the specification for the To-Do List Application using Data Flow Diagrams that explains data flow through the system, including processes, data stores, and external entities.

2. System Context

2.1 External Entities

The following external entities interact with the system:

User

External Authentication Provider

Email Service

Calendar Integration

Mobile/Web Client

3. Data Flow Diagram Levels

3.1 Level 0 (Context Diagram)

```
`

+---------------------+

          | To-Do List      |

|    System         |

             +------------+

/         |          \

/          |           \

User Input ---           |            --- System Output

Authentication ---     |              --- Notifications

Task Data ------       |             --- Sync Data
```

Task Data Elements:

Task ID

Title

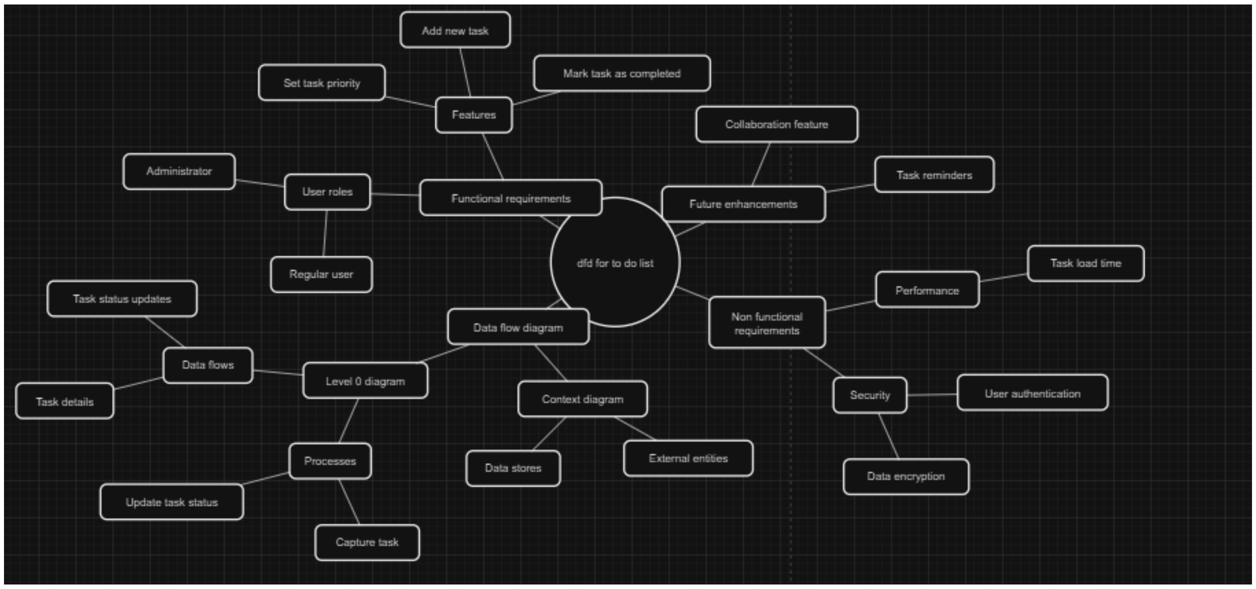
Description

Due Date

Priority

Status

Assigned User



Swimlane Diagram Specification

1. Introduction

1.1 Purpose

This document specifies the swimlane diagram for the To-Do List Application. The swimlane diagram illustrates the interactions between various system components and external entities by categorizing actions into distinct lanes, representing roles or subsystems.

Swimlane Diagram Elements

Roles/Subsystems:

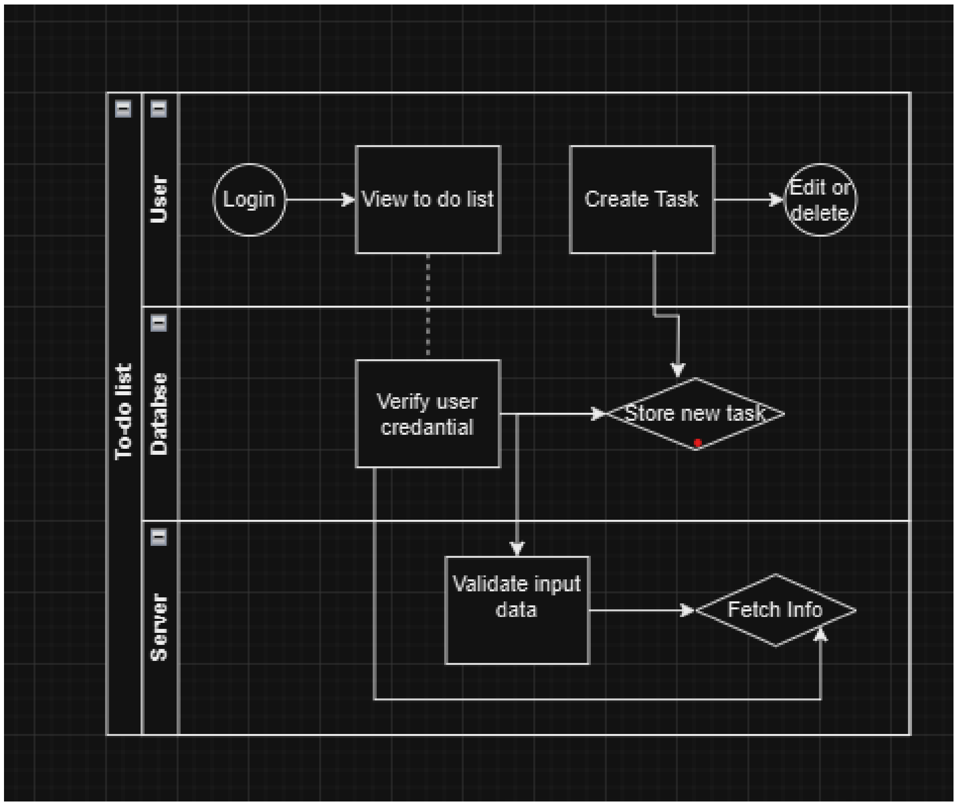
1. **User**: Primary actor interacting with the application.
2. **Application Backend**: Manages business logic and acts as an intermediary between the user, database, and external services.
3. **Database**: Central repository for task data.
4. **External Services**: Includes authentication, email notifications, and calendar sync providers.

Processes:
Each swimlane represents a specific entity or role, and arrows indicate the flow of information and control between processes.

Swimlane Diagram Representation

The swimlane diagram visually represents:

1. **Vertical Lanes**: For each entity or role (User, Backend, Database, External Services).
2. **Horizontal Flow**: Shows sequential steps and their interaction across lanes.


To-do list
User
Databse
Server
Login
View to do list
Create Task
Edit or delete
Verify user credantial
Store new task
Validate input data
Fetch Info