TimeWise

SPL-02 Project Report

Submitted By

Abdus Salam Islam Badhon (BSSE 1401)

Md. Nahidur Rahman Nahid (BSSE 1429)

Supervised By

Mridha Md Nafis Fuad

Lecturer, IIT, DU

_____

Supervisor's Signature

Table of Contents

Introduction:

Welcome to the Software Requirements Specification(SRS) for TimeWise, a productivity management tool through task management. This document highlights the functionalities and requirements of the tool. This document will firstly begin by addressing the inception and elicitation report of TimeWise.Then it will address different modelings like scenario based modelings, database modeling, class based modeling , behavioral modeling . Then finally this document will address the data flow diagram. Finally we will have the user manual to help users navigate through the site.

Purpose and Motivation:

The main purpose of TimeWise is to help users manage their tasks in a proper way to become more productive. In this era of social media distractions, a tool that organizes daily tasks with proper timings and deadlines along with time to time reminders can lessen procrastination and can make daily activities more manageable.

Intended Audience:

All the reports,modeling and diagrams used in this SRS can give a proper and clear idea about our tool which can be helpful for software

developers,stakeholders,system architects, project managers, quality assurance professionals,end users as well as someone who is interested about the functioning of TimeWise.

Conclusion:

In conclusion we the creators of the tool Abdus Salam Islam Badhon (bsse1401@iit.du.ac.bd) and Md Nahidur Rahman Nahid (bsse1429@iit.du.ac.bd) believe that this SRS will help the intended audience to get the clear understanding of our tool. And we are also grateful to our supervisor Mridha Md Nafis Fuad(fuad@iit.du.ac.bd) sir for guiding us regarding the project.

1. Inception Report:

1.1 Project Overview and Objectives

TimeWise is a productivity tool that helps users to manage their daily activities.. Our tool now totally free to use so this document will not address any payment related specifications.TimeWise will have different features like listing daily activities in a to-do format, tracking progress and performance, getting time to time reminders and notifications, seeing relevant statistics of the participations,getting personalized feedback and tips and managing productivity in a collaborative and competitive environment.

1.2 Stakeholders

TimeWise is a tool which can be useful for anyone trying to organize their task and time. It is not created based on a particular group of people like students, working professionals,employees,managers and so on. It can be useful for many categories of people. So in the rest of the documents we will address all of these categories of people simply as users.So the stakeholders are the followings:

1.Users
  1.1 Students
  1.2 Teachers
  1.3 Working Professionals
  1.4 Employees

1.5 Managers
1.6 Others

1.3 Methodology and Requirements Collection Plan

To collect requirements we will talk to the stakeholders personally and will collect their personal experience and recommendations regarding managing their daily activities. We may also collect information from a vast amount of stakeholders by some online means. We will also refer to research papers, reports,relevant books for collecting information about productivity related contents. We will take notes about our findings and will refer to these in developing and planning the system.

1.4 Assumptions and Risk

We will assume that the user will behave in a fair way like giving the actual input. If they dont the outcomes and results that our system will show will be biased and will be not beneficial for the users. Our current system will not address any dedicated efforts to stop cheating of the user. Maybe in the later versions we may consider these issues. So we expect the user to be fair for their own benefits.

1.5. Timeline and Strategy

In the beginning phase we will be collecting requirements and relevant information and make proper documentation of the findings. Then the UI/UX planning and actual development will be done. Followed by testing and bug fixing. Finally we will deploy the tools and will provide user documentation for better user experience. We will follow the iterative process model to develop the actual system. Where after each iteration we will perform the required changes. And we may develop independent parts parallel so that integration of different modules becomes easy.

2. Elicitation Report:

2.1 Quality Function Deployment(QFD)

The Quality Function Deployment(QFD) is a structured approach to define user needs or requirements and translating them into specific plans to produce products to meet those needs.Based on our observations after interaction and research on the subject we summarized the following requirements that will be implemented on our tool.

2.1.1 Normal Requirements

1. Users Account management: The user will be able to create an account by providing some personal information like name,email address and so on.Our system will maintain a profile for each user.
2. Task Management: The user will maintain some tasks which they will perform from time to time. The task will contain a task goal which will help the user cluster the taks with a broader sense.
3. Reminders & Notifications: Users will be notified from time to time based on his setted tasks and activities.

2.1.2 Expected Requirements

1. Progress and Performance Report : Based on users tasks, the user will be able to track his progress of the current assigned tasks.Our system will provide a daily progress report which will contain the deadline and current progress of all the unfinished tasks. And users can see this report at any time and based on some previous number of days consideration.
2. Showing Stats: Users will be able to see relevant statistics about his performance and participation. Stats will be shown on multiple categories like task, account,feedback and session.
3. Messages & Feedbacks: Users will be able to send and receive messages and feedback from other users. Feedback will contain the task name and the feedback score.

2.1.3 Exciting Requirements

1. Session Management: Users will be able to create a session by defining session goals and can start a stopwatch. In this session the user will perform tasks which he includes in the operated tasks list and after the session completion the user will defiance the session outcome and session efficiency which will be used in the generating deep work analytics.

2. Deep work analytics: Users can track their hours spent on sessions, no of tasks operated within these sessions and also an average productive score over the different hours of the day in the form of productivity meter.

3. Collaborative and competitive environment: Users will be able to share their tasks and plans with others. They can also create a team and share tasks with team members.Users can also send messages and can see other users' accounts .

3. User story:

3.1 User Registration and Profile Management

User wants to create an account and manage his profile, so that he can securely use TimeWise and access personalized features.

- Scenario:
    1. The user accesses the registration page.
    2. The user enters their name, email,role and other required information.

3. The system validates the input(unique username) and sends account verification code via email. User enters this and completes his registration. User can log in using username and password and if he forgets his credentials he can recover his account by giving email address and getting the account recovery code in the email. The user logs in to access their personalized dashboard.
4. The user updates their profile with additional preferences, such as account visibility status(private or public) and activity status (active or inactive).User name can be changed or updated and unique within the system.
5. Users can follow others and can see other users' profiles also and can see(if public) what tasks they are participating or what teams they joined.

3.2 Task Management

User wants to maintain a list of tasks so that he can organize his goals and track their completion.

Scenario:

1. The user logs into the TimeWise dashboard.
2. The user creates a task and adds some todo under this task.Task contains multiple attributes like task name, description, priority level, deadline and so on. Users can share a task with others and can collaboratively perform the tasks. Each task has to have at least one task to do and the task completion and progress depends on this to do completion (done or undone).User can comment on the task that is visible to everyone and also can take personal note specific to that task.

3. The system displays a list of tasks with filters for priority, deadlines, and categories.Sorting and filtering tasks based on different criteria can help the user to choose what task to perform first and what next.
4. The user mark's task to be completed and based on that the task current progress is updated . Once all the tasks are marked as done the task completion status reaches 100 percent and gets completed.
5. The user can comment on the task which is visible to all the task participants and also can take note of the task which is personal only to that user only.
6. Any changes in the task like task details or task todo status a log is stored about the change. And the user can see the task modification history of the task to get acknowledgement about all the changes happening over the time on the task, especially the task progress status and who made those progress to be changed and the status of the changes.

3.3 Reminders and Notifications

As a user, I want to receive reminders and notifications for my tasks,so that I can stay on track and meet deadlines.

- Scenario:
    1. The system sends any kind of events like a task or team invitation, a task assignment in the team, a participated task deletion. Also these notifications are sent while registration and account recovery process to share verification code.
    2. The user can view his previous notifications from the notifications dashboard. He can also sort and filter the notifications based on different categories like invitations, deletion, response and so on. He can remove the notifications also which will be removed from the panel.

3.4 Progress and Performance Tracking

User wants to track his progress and evaluate his performance,so that he can see how effectively he is achieving his goals.

- Scenario:
    1. The user gets the progress report daily twice about the unfinished tasks. This unfinished task can be crossed or before the deadline.
    2. The users get the performance report once a week and this report contains different statistical data about the performance of the user on tasks feedback season and the account as a whole.
    3. Apart from the system generated report the user can see the report any time within the site and for performance report he can set a particular previous number of days activity to consider to get more relevant information about the performance.

3.5 Statistics and Insights

As a user,I want to see statistics about my productivity,so that I can analyze and improve my daily performance.

- Scenario:
    1. The system generates visualizations such as bar graphs and pie charts showing different kinds of stats related to tasks, sessions, feedbacks, and account related stats.
    2. The user can filter statistics by some previous number of days.
    3. Different visuals appropriate for a particular kinds of data and also in some situations multiple visualizations to show different statistical results may enhance the understanding of the stats and may help in getting better understanding of the stats

3.6 Messages and Feedbacks

As a user,I want to send and get messages and feedback from other users.

- Scenario:
    1. The User can draft a message and feedback and can send it to the other user. The recipient will get an email containing this message and feedback.
    2. In the feedback the user can attach the task name on which they are giving the feedback. He can also attach a feedback score which will be a form of evaluation of the task.
    3. Users can remove the messages and feedback form the corresponding message and feedback dashboard.

3.7 Session Management

User wants to perform a working session where the user wants to perform certain tasks.

Scenario:

1. User creates a session by defining the session goal,duration , and a preferred environment or background to be opened at the session .
2. During the session users can take notes which will be preferred as session outcome. Users can also update the list of tasks they are operating. They can see the timer of the session to keep track of the progress and timing.
3. If a user closes the tab or moves to some other route within the site the session stats are kept saved so that when they come back they can resume the session from where they left instead of creating a new session again.

4. After completing the session the user can save the season details or may not. Before saving the session he can put an efficiency score featuring the effectiveness and productivity measure of the session which will be used later on in getting different insights.
5. Users can see his previous session details to keep track what he did in previous sessions and what was their outcome and goal.

3.8 Deep Work Analytics

As a user,I want to monitor my deep work sessions,so that I can identify the time periods when I am most productive.

- Scenario:
    1. The user can see different time distributions that he spent on the sessions on the site along with their efficiency scores.
    2. For more personalized results the user can set a previous number day's consideration and only on those days session will be considered giving the user more accurate and fine tuned results.
    3. Different methods are there timely (Day, Evening,Night) or hourly basis can be set to see the results.
    4. Different indicators and insight are there to show productive measures giving users more accurate and precise deep working analytical results.

3.9 Collaborative and Competitive Environment

As a user,I want to share my task with others,so that I can collaborate and compete to stay motivated.

- Scenario:
    1. The user can create a team and add team members.
    2. Only a team owner will be able to add tasks within a team,invite others to the team or accept requests to join the team and change

team details . But team names can be changed and have to be unique within the system.

3. Once a task is assigned in a team all the team members get notified and get access to the task. And if a task is removed from the team all the team members lose access to that removed task.If a user left a team then he loses access to all the team tasks participation.
4. Any event on the team like team members' addition or removal , changing team details is kept at the team history. Which the team members can see.
5. Users can perform chatting in a team chat box to share their opinions and ideas.

4. TimeWise Labs

.TimeWise Labs is our latest extension to TimeWise.The main motive of this section is to use **Artificial Intelligence(AI)** to help automating manual tasks, giving insights, generating content to increase the productivity of task management and its related activities. Following are some of the ways AI has been used within the TimeWise:

4.1 Generating Roadmap: To create a task or set or task the user had to manually set the task name and its corresponding task todos which can be difficult especially for creating a task for which the user don't have any prior knowledge like setting tasks for learning a new language or organizing a sports tournament. So the user can set a goal for a roadmap optionally he can also set other parameters like deadline, no of tasks or extra prompt to regain the roadmap generation and our system based on that with the help of AI will generate a set of tasks to complete the roadmap with the defined goal. After getting the generated task the user can modify the task details and then can save them in the database and perform these tasks just like other tasks(sharing with team, updating task status and so on).

4.2 Autocompleting Message and Feedback: To draft and write a message or feedback to others it can be difficult to sum up or summarize a concept or idea . So the user can just pass the message subject and our system will generate a compact and insightful message or feedback which the user can revise and update and then send to the other user.

4.3 Creating A Session Outline: During creating or performing a session a user may not have a clear idea or clue what task should they perform or perform first and also what should be the goal and outcome of the session . For that our system will analyze the user's existing tasks like tasks priority deadline current progress and will create a session outline containing session goal outcome and set of tasks that the user can perform in that session . This can help the user to focus on the task to perform and things to do rather than what to do .

4.4 Analyzing and Summarizing Contents: The user may have a lot of messages or feedback in the inbox manually checking them can be boring and inefficient. And also all the contents like task profiles, previous session details, stats and deep work analytical results manually seeing them and getting the insights or actions to perform may be time consuming or difficult. So the user can tap the analyze button. Our system will analyze content and give insight and summarize results. While getting the result the user can set different analysis types like concise, standard, general and so on . along with that as a prompt the user and set the context based on which the user will get more precise and personalize results.

5. Scenario Based Modeling

The success of a computer-based system depends highly on the user's satisfaction. So if we can create different scenarios of users interacting with our system we can target these interaction scenarios and make them useful to give an overall good experience. Scenario based modeling highlights these users' interactions. This modeling uses different diagrams like use case diagram, activity diagram and swimlane diagram.These diagrams along with proper description and explanation are given below:

5.1 Use Case Diagram: Timewise

A use case is a detailed description of how a user interacts with a system to accomplish a specific goal. It focuses on the user's perspective, outlining the steps involved. Primary actors are the main users directly interacting with the system and positioned at the left side , while secondary actors are external entities that provide or receive information but aren't directly using the system positioned at the right side.In the middle the main modules with which the user interacts are placed.Actors doesnt need to be physical person it can be intangible entity also like mail,third party software and so on.To show the interaction between modules and actors a line is drawn between actors and modules name. Now the use case diagram of TimeWise will be shown below:

5.1.1 Level 0

Name: TimeWise(A time and productivity management system)
PrimaryActor:
User(Student,Teacher,WorkingPrefessional,Employee,Managers,Others)
Secondary Actor: Central Database,Notification System

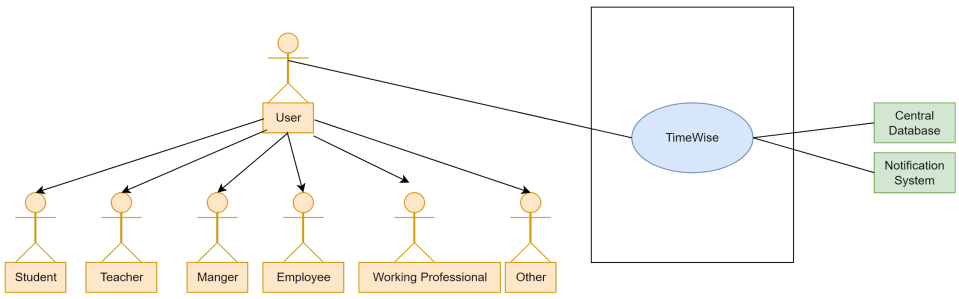


Figure 1: TimeWise(A time and productivity management system)

5.1.2 Level 1

1. **Name:** TimeWise System Details

PrimaryActor:
User(Student,Teacher,WorkingPrefessional,Employee,Managers,Others)
Secondary Actor: Central Database,Notification System,Analytics Engine,Collaboration Engine,Statistics Engine

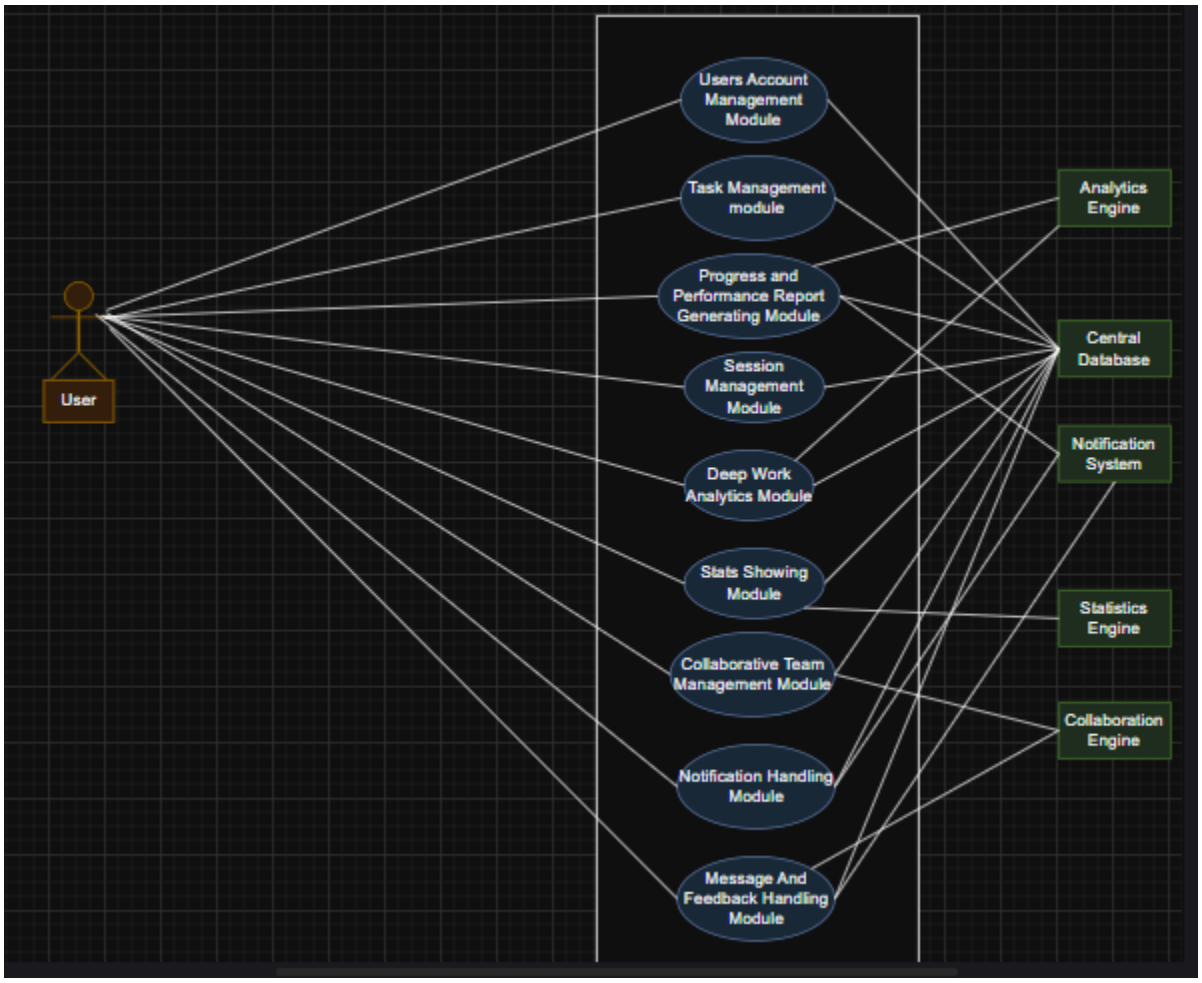


Figure 2: TimeWise System Details

5.1.2.1 Level 1.1

Name: Users Account Management Module
PrimaryActor:
User(Student,Teacher,WorkingPrefessional,Employee,Managers,Others)
Secondary Actor: Central Database

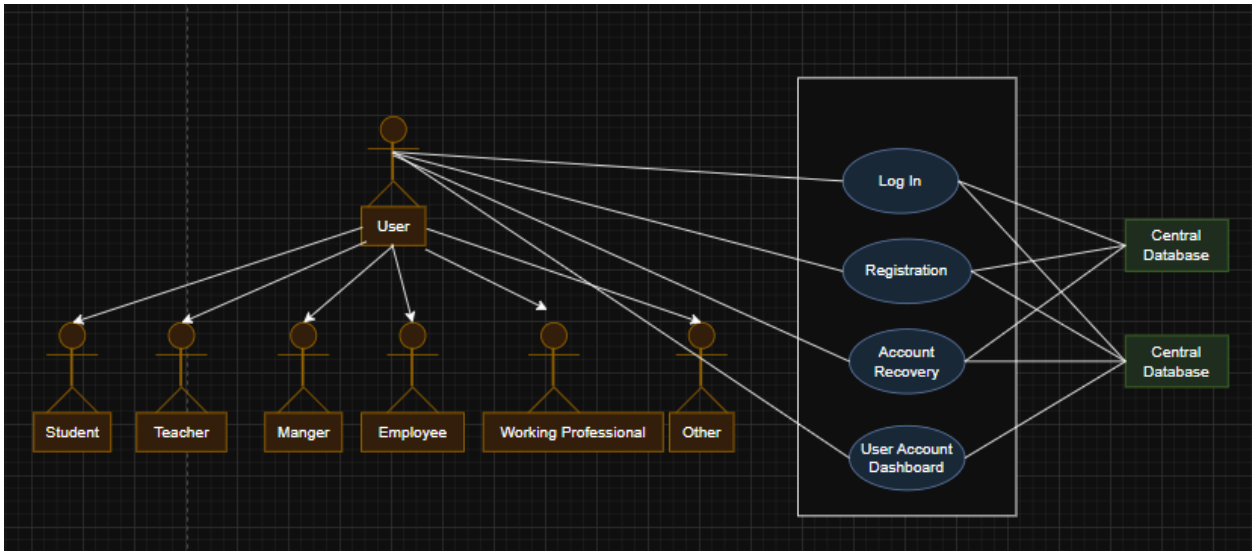


Figure 3: User Account Management Module

Description:
Users will be able to login and registration by providing necessary credentials. They can manage their account by changing activity status or log out.their account. Users will also be able to modify their profile details and can see all of their information in the user dashboard. They can see their account related stats

and can view other users' profiles and their operated tasks and participating teams.

5.1.2.2 Level 1.2

Name: Task Management Module

PrimaryActor:

User(Student,Teacher,WorkingPrefessional,Employee,Managers,Others)

Secondary Actor: Central Database,AI service,Collaboration engine,Statistics Engine

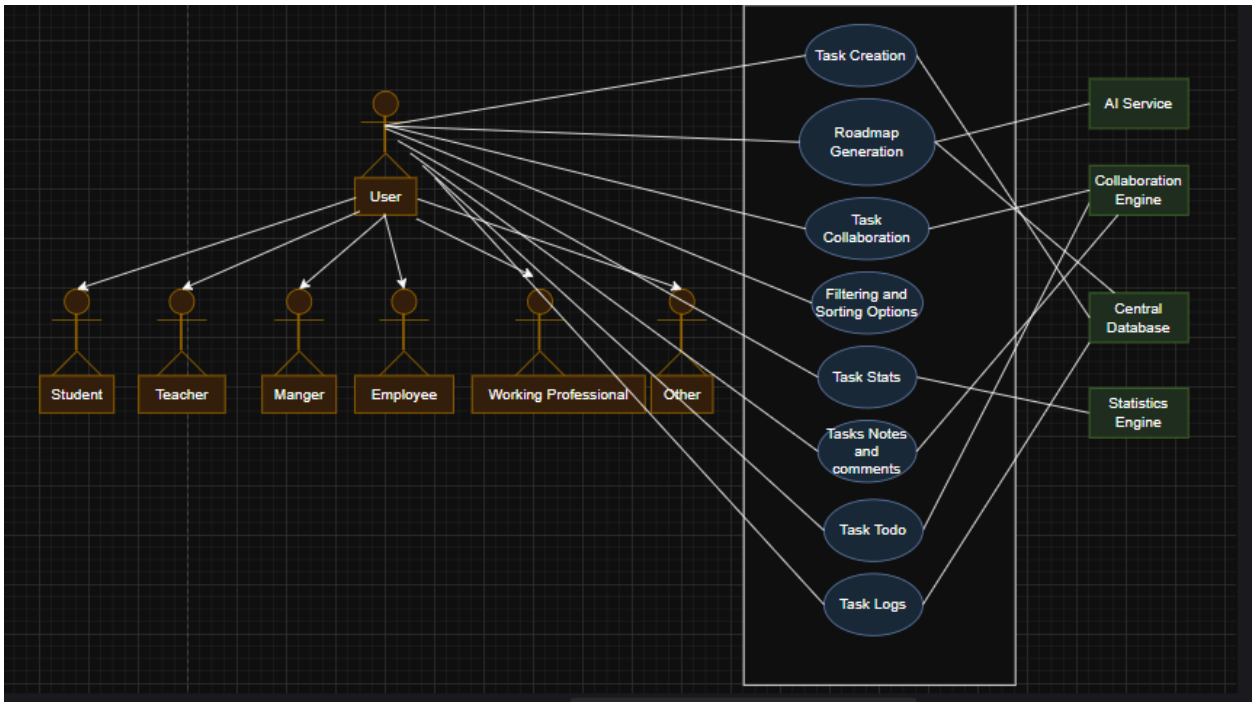


Figure 4: Task Management Module

Description:

Users will be able to manage tasks like adding,updating tasks details. They can set and modify the details about task prioritization and completion. They can categorize,filter,sort and tag different tasks.They can share tasks with others, generate tasks through roadmap generation, add notes and comments,add and update task todo status.

5.1.2.3 Level 1.3

Name: Progress And Performance Report Generating Module
PrimaryActor:
User(Student,Teacher,WorkingPrefessional,Employee,Managers,Others)
Secondary Actor: Central Database,Statistics engine,Collaboration engine,Analytics engine

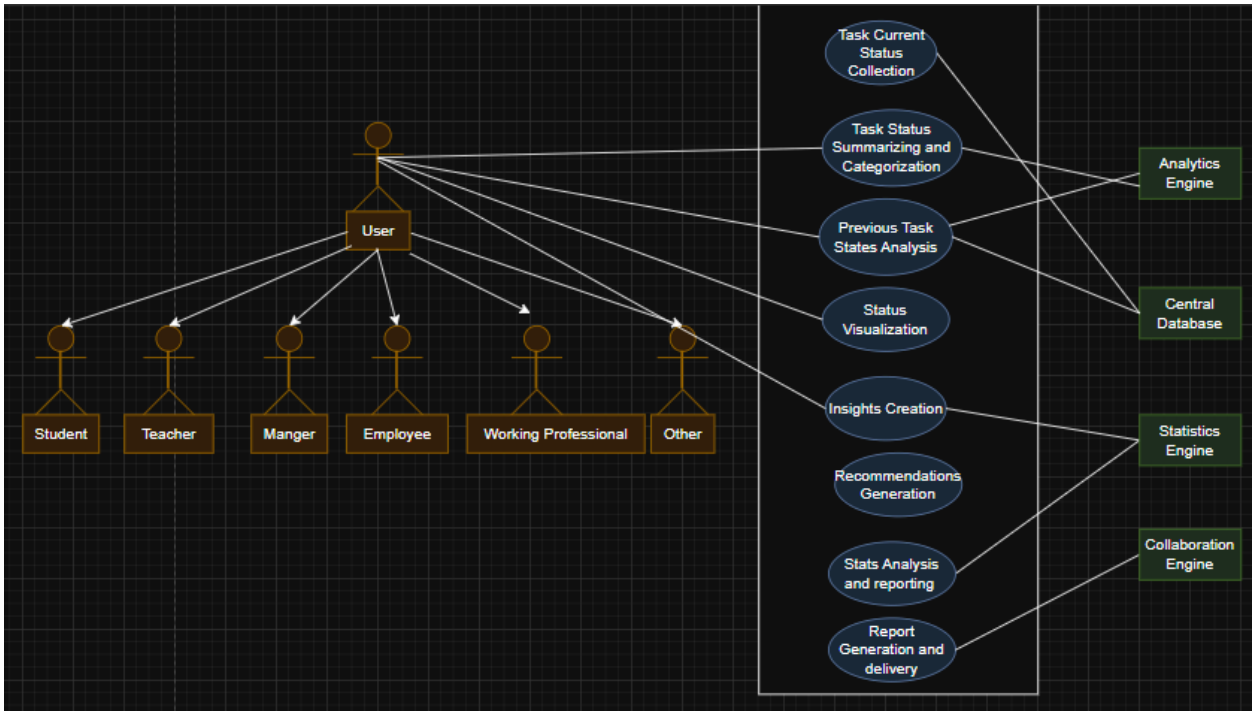


Figure 5: Progress And Performance Report Generating Module

Description:
Users can monitor progress status like seeing deadlines and completion status which can help them to focus on the tasks that are close to the deadline but not yet finished. They see the progress reports to work on the ongoing task.

5.1.2.5 Level 1.5

Name: Reminders and Notification Module
PrimaryActor:
User(Student,Teacher,WorkingPrefessional,Employee,Managers,Others)
Secondary Actor: Central Database,Email Service

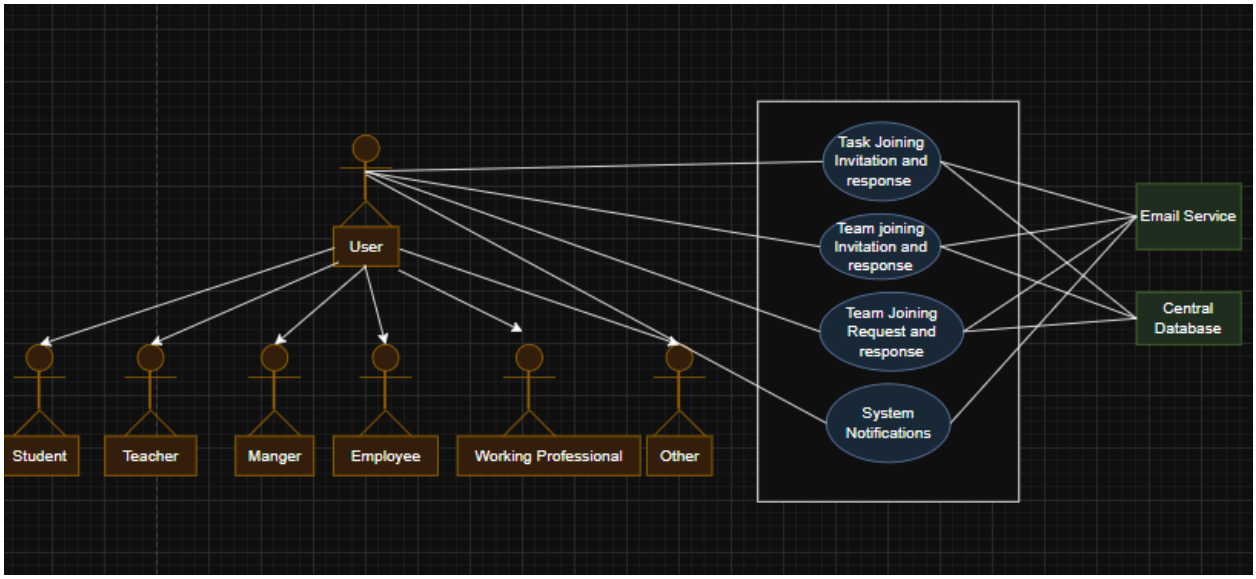


Figure 7: Reminders and Notification Module

Description:

Users will get time to time alerts to deadlines and unfinished tasks. They will also get general notification regarding the service. Based on priority of tasks they will also get notification.

5.1.2.6 Level 1.6

Name: Stats Showing Module
PrimaryActor:
User(Student,Teacher,WorkingPrefessional,Employee,Managers,Others)
Secondary Actor: Central Database,visualization service,statistics engine

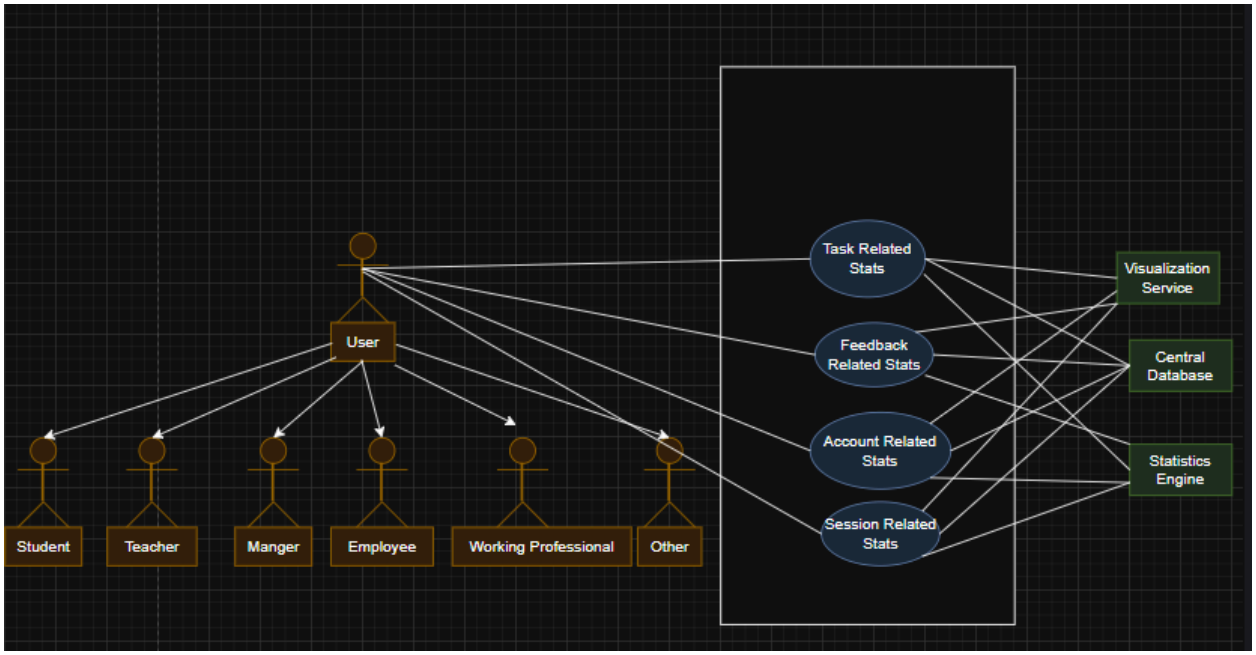


Figure 8:Stats Showing Module

Description:

The statistics engine with the help of a central database will visualize the user performance with the help of relevant statistics.

5.1.2.7 Level 1.7

Name: Deep Work Analytics Module
PrimaryActor:
User(Student,Teacher,WorkingPrefessional,Employee,Managers,Others)
Secondary Actor: Central Database,Analytics Engine

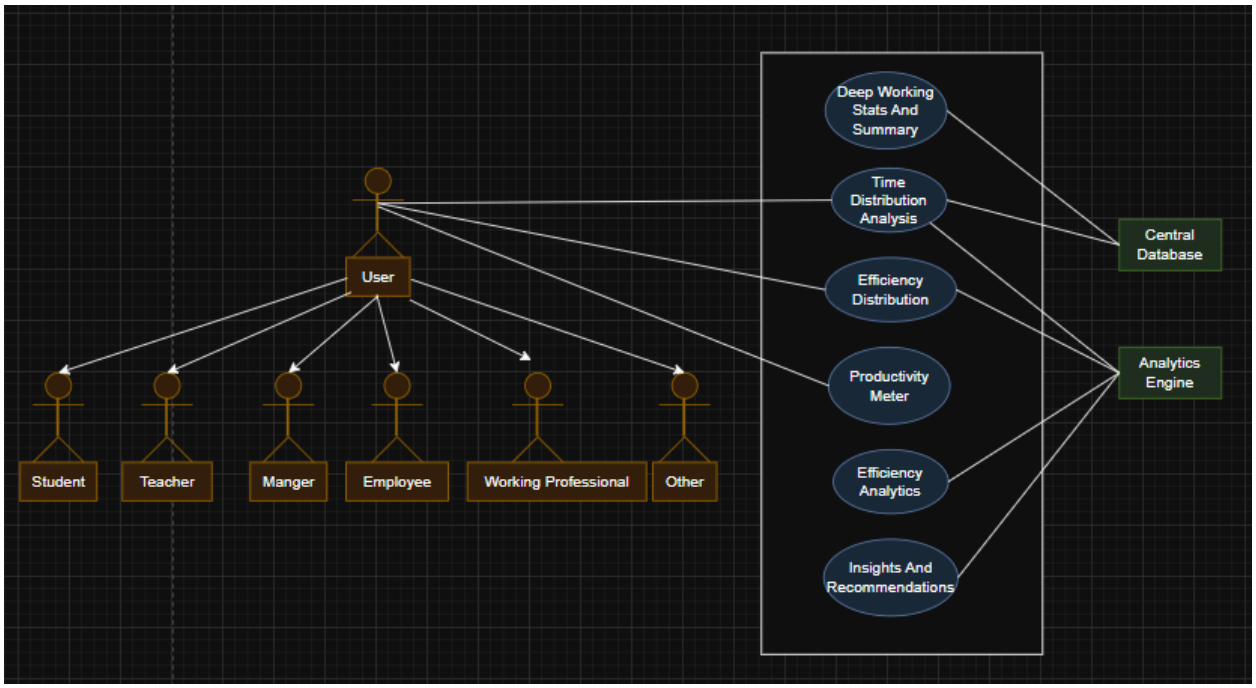


Figure 9: Deep Work Analytics Module

5.1.2.8 Level 1.8

Name: Collaborative and Competitive Environment Creating Module

PrimaryActor:
User(Student,Teacher,WorkingPrefessional,Employee,Managers,Others)
Secondary Actor: Central Database

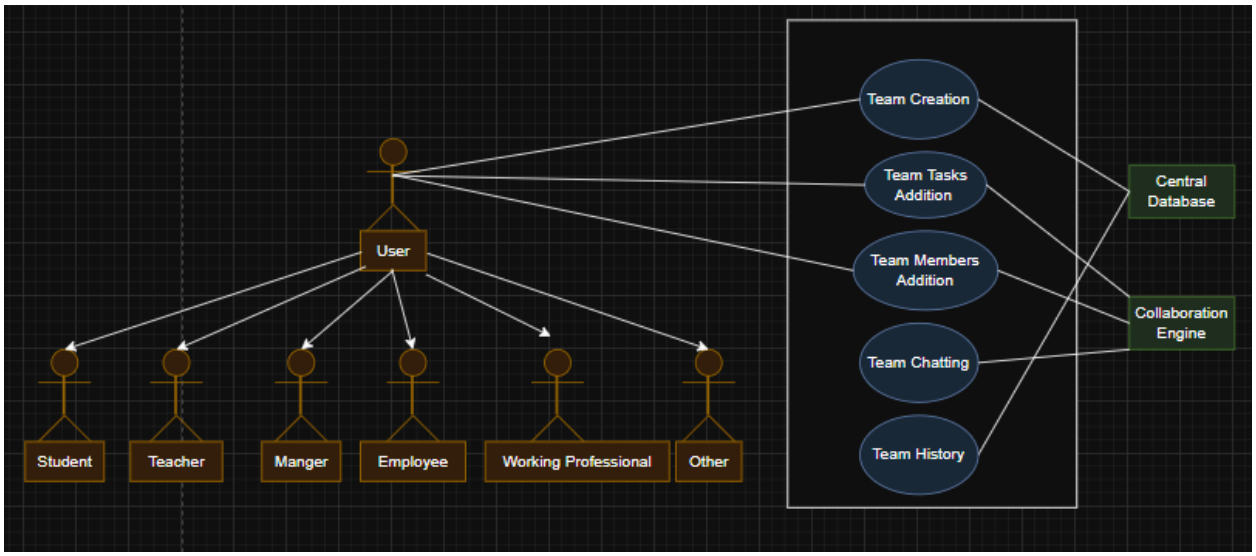
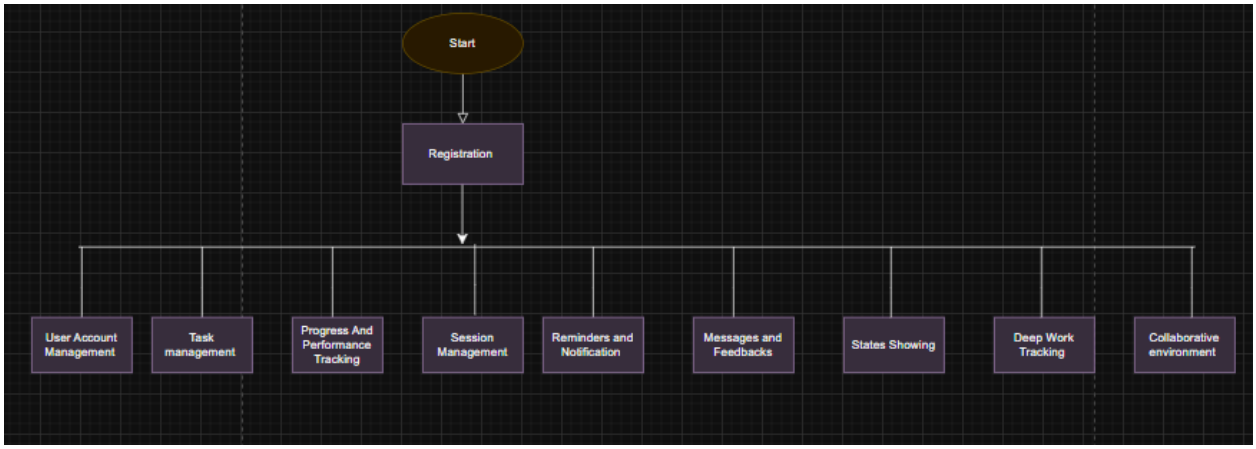


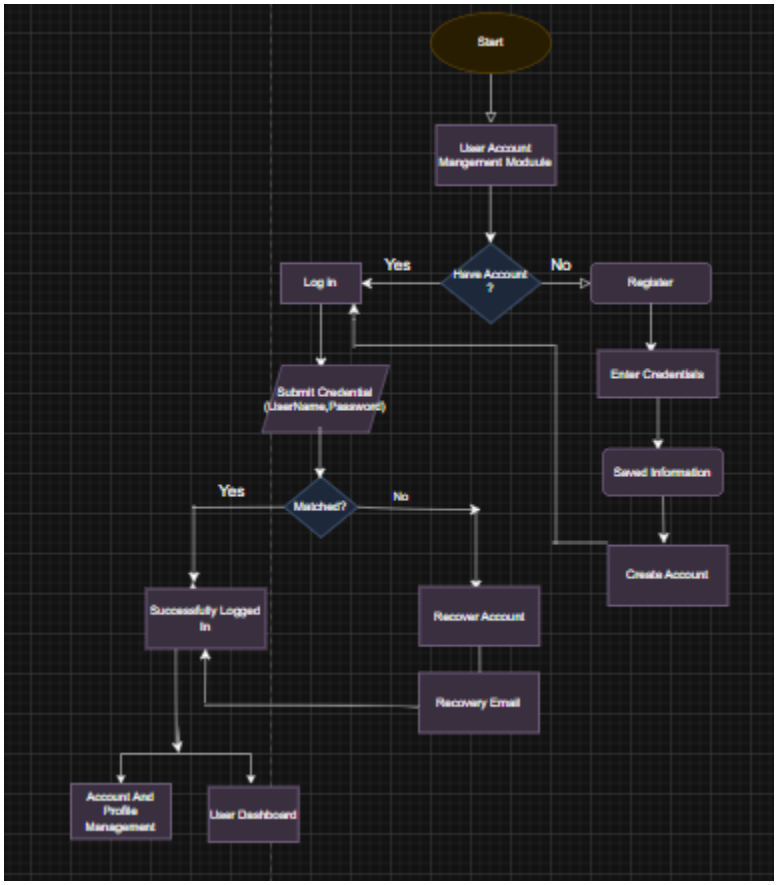
Figure 10: Collaborative and Competitive Environment Creating Module

Activity Diagram: TimeWise

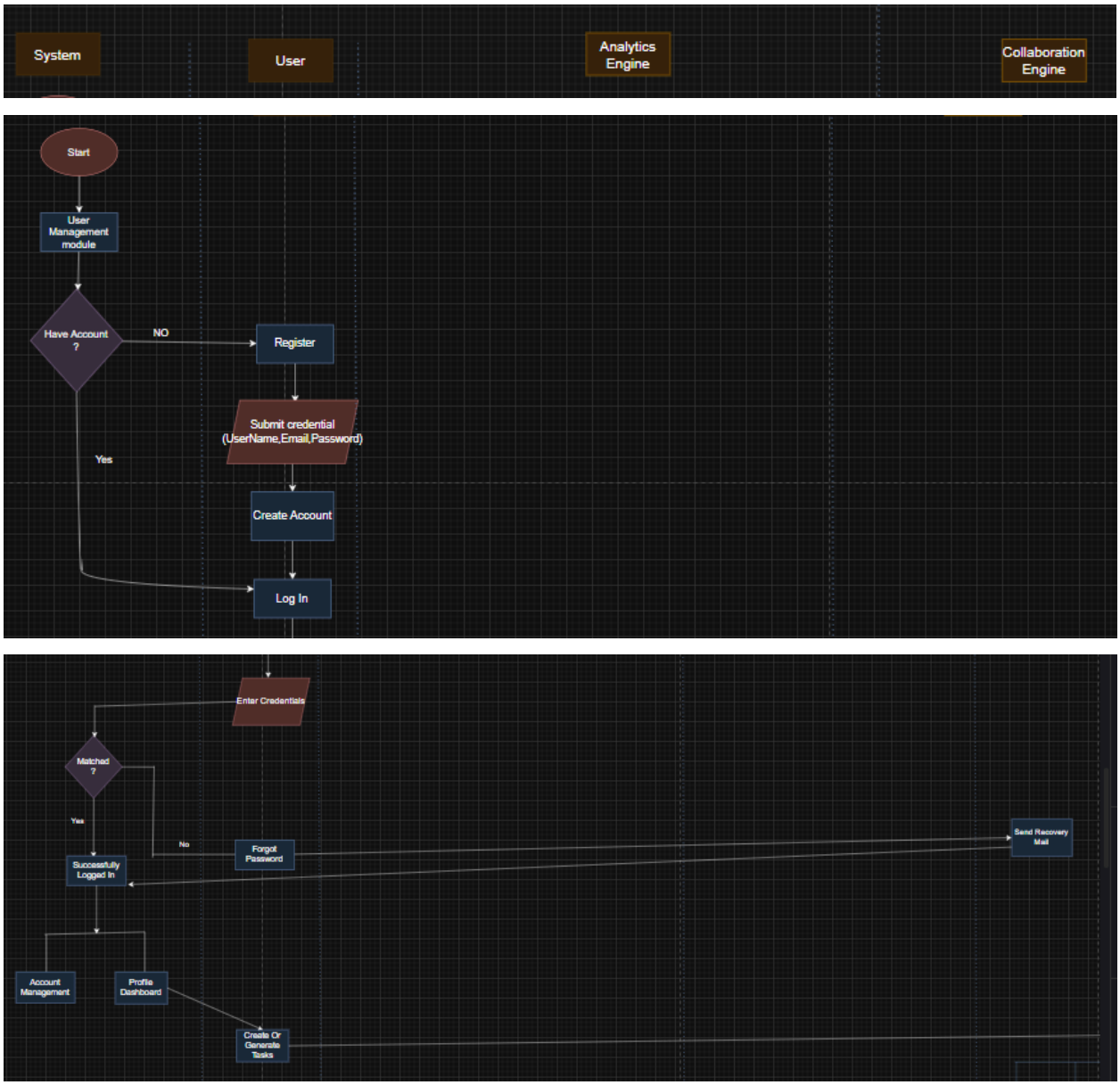
Level 1:

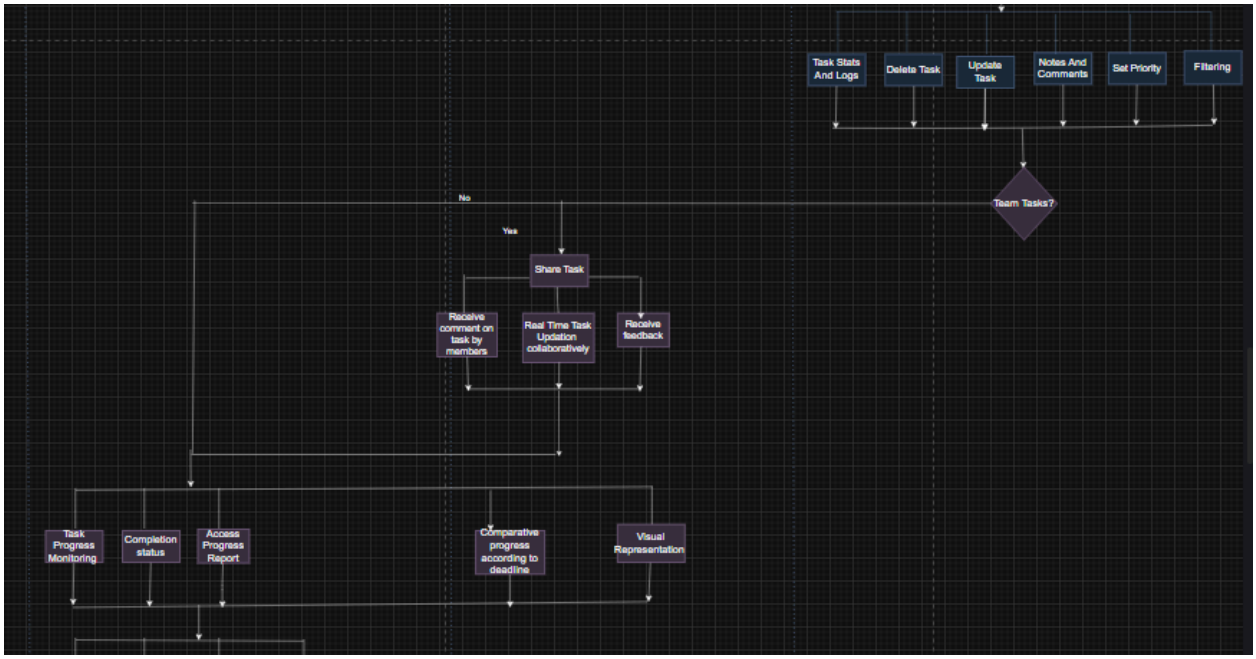

Start
Registration
User Account Management
Task management
Progress And Performance Tracking
Session Management
Reminders and Notification
Messages and Feedbacks
States Showing
Deep Work Tracking
Collaborative environment

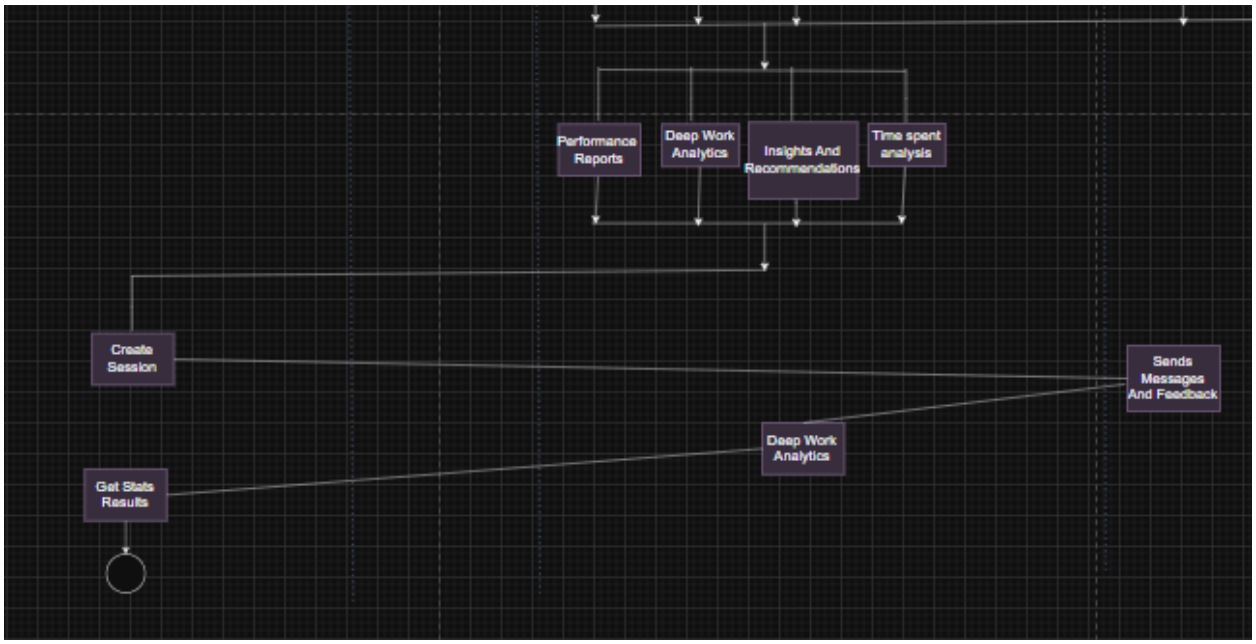
Level 1.1:


Start
User Account Mangement Module
Yes
Have Account ?
No
Log in
Register
Enter Credentials
Submit Credential (UserName,Password)
Saved Information
Yes
Matched?
No
Create Account
Successfully Logged In
Recover Account
Recovery Email
Account And Profile Management
User Dashboard

3. Swimlane Diagram: TimeWise







4. Database Modeling

4.1 What is Database Modeling?

Database modeling is the process of designing a detailed data structure that defines how information will be stored, organized, and accessed within a database. It involves creating conceptual, logical, and physical models that represent data relationships, constraints, and flow.

4.2 ThePurpose of Database Modeling

The purpose of database modeling is to ensure the database is optimized for storing, retrieving, and managing data efficiently. It helps maintain data integrity, consistency, and accessibility while avoiding redundancy and supporting the application's functionality.

4.3 Database Modeling: TimeWise

4.3.1 Noun Listing

No.	Nouns	Attributes (Related Nouns)	Problem Space(P)/ Solution Space(S)
01	User	2, 3, 4, 5, 6, 7, 24, 25	P
02	Student	1, 3, 4, 5, 6, 8, 13, 14, 21, 35, 36	P
03	Teacher	1, 2, 4, 5, 6, 8, 16, 32, 14, 15	S
04	Manager	1, 2, 3, 5, 6, 8, 14, 16, 31, 34	P

No.	Nouns	Attributes (Related Nouns)	Problem Space(P)/ Solution Space(S)
01	User	2, 3, 4, 5, 6, 7, 24, 25	P
02	Student	1, 3, 4, 5, 6, 8, 13, 14, 21, 35, 36	P
05	Employee	1, 2, 3, 4, 6, 8, 16, 31	S
06	Working Professional	1, 2, 3, 4, 5, 8, 10, 11, 12, 13	S
07	Others	1, 2, 3, 4, 5, 6, 8, 13, 16, 36	S
08	Task	2, 3, 4, 5, 6, 7, 21, 26, 27	S
09	Progress	2, 5, 8, 10, 37, 40, 52	P
10	Performance	2, 5, 6, 8, 14, 19, 37, 38	P
11	Notification	6, 12, 15, 23, 24	P
12	Reminder	6, 11, 29, 43	P
13	Stats	2, 8, 36, 37, 50	S
14	Goal	2, 3, 4, 8, 36	S
15	Team	1, 3, 4, 32, 47	P
16	Feedback	3, 5, 8, 38, 47	S
17	Feedback Type	16, 18	S
18	Role	1, 2, 3, 4, 5, 6	S
19	Category	2, 8, 37, 51	S

No.	Nouns	Attributes (Related Nouns)	Problem Space(P)/ Solution Space(S)
01	User	2, 3, 4, 5, 6, 7, 24, 25	P
02	Student	1, 3, 4, 5, 6, 8, 13, 14, 21, 35, 36	P
20	Session	1, 2, 3, 5, 45	S
21	TimeWise	1, 2, 3, 5, 8, 36	S
22	LogIn	1, 23, 24	P
23	Register	1, 24, 25	P
24	Account	1, 2, 3, 5, 23	P
26	Deadline	8, 27, 43	P
27	Priority	8, 26, 30	S
28	Comments	8, 32	S
29	Timer	12, 43	S
30	Productivity Meter	2, 10, 38, 44	P
31	Task Sharing	4, 8, 15, 47	P
32	Task Administration	3, 4, 8, 15, 25	P
33	Analytics Engine	13, 38, 41, 52	S
34	User Dashboard	1, 21	S
35	Date and time service	2, 8, 21, 26, 29	S

No.	Nouns	Attributes (Related Nouns)	Problem Space(P)/ Solution Space(S)
01	User	2, 3, 4, 5, 6, 7, 24, 25	P
02	Student	1, 3, 4, 5, 6, 8, 13, 14, 21, 35, 36	P
36	Goal tracking	2, 8, 14	S
37	Progress Analytics	9, 36, 44	P
38	Performance Analytics	10, 16, 37	P
39	Reporting service	33, 41, 52	S
40	Task Modification logs	10, 24	P
41	Statistics engine	13, 38, 39	S
42	Visualization service	9, 13, 33	S
43	Deep Work	29, 30	S
44	Efficiency analytics	2, 10, 38	S
45	Session Stats	20, 37	S
46	Task status	8, 31	S
47	Collaboration engine	1, 4, 15, 31	S
48	Central database	24, 40	P
49	Task completion stats	8, 13, 50	S

No.	Nouns	Attributes (Related Nouns)	Problem Space(P)/ Solution Space(S)
01	User	2, 3, 4, 5, 6, 7, 24, 25	P
02	Student	1, 3, 4, 5, 6, 8, 13, 14, 21, 35, 36	P
50	Time spent metrix	8, 13, 44	P
51	Category based stats	19, 37	S
52	Progress reports	9, 33, 39	S
53	System Services	33, 42, 47	S

4.3.2 Collections Schema

User

{

```
  "userId": "ObjectId",
  "userName": "String",
  "userEmail": "String",
  "password": "String",
  "shortBioData": "String",
  "role": "String",
  "userStatus": "String"  "usersFollowing": ["String"]
}
```


Task


```
{
  "taskId": "ObjectId",
  "taskName": "String",
  "taskCategory": "String",
  "taskDescription": "String",
  "taskPriority": "String",
  "taskVisibilityStatus": "String",
```

```
"taskCreationDate": "Date",
"taskDeadline": "Date",
"taskOwner": "String",
"taskGoal": "String",
"taskParticipants": ["String"],
"invitedMembers": ["String"],
"taskComments": [
  {
    "timestamp": "Date",
    "userName": "String",
    "commentText": "String"
  }
],
"taskNotes": {
  "String": [
    {
      "timestamp": "Date",
      "noteText": "String"
    }
```

```
  ]
},
"taskCurrentProgress": "Integer",
"taskModificationHistory": [
  {
    "timestamp": "Date",
    "fieldName": "String",
    "updatedBy": "String",
    "previousValue": "Object",
    "newValue": "Object"
  }
],
"taskTodos": [
  {
    "timestamp": "Date",
    "description": "String",
    "status": "String"
  }
]
```

}

Progress Report

```
{
  "progressReportId": "ObjectId",
  "userName": "String",
  "taskStatuses": [
    {
      "taskName": "String",
      "taskOwner": "String",
      "taskPriority": "String",
      "tasksCurrentProgress": "Integer",
      "taskCreationDate": "Date",
      "taskDeadline": "Date",
      "isDeadlineCrossed": "Boolean"
    }
  ],
  "timeStamp": "Date"
}
```

Performance Report


```
{
  "performanceReportId": "ObjectId",
  "userName": "String",
  "usersAccountStatistics": {
    "numberOfUserFollowing": "Long",
    "teamsParticipated": "Long",
    "numberOfTasksParticipated": "Long",
    "numberOfSessionsCreated": "Long",
    "previousFeedbackScores": ["Integer"],
    "numberOfMessagesSent": "Long",
    "numberOfMessagesReceived": "Long"
  },
  "usersTaskStatistics": {
    "totalTasksCreated": "Long",
    "totalTasksCompleted": "Long",
    "totalTasksInProgress": "Long",
```

```
  "averageTaskCompletionTime": "Double",
  "taskPrioritiesDistribution": {
    "high": "Long",
    "medium": "Long",
    "low": "Long"
  }
},
"usersSessionStatistics": {
  "numberOfSession": "Integer",
  "totalSessionTime": "Double",
  "averageSessionEfficiency": "Double",
  "totalTasksOperated": "Integer",
  "sessionNames": ["String"]
},
"usersFeedbackStatistics": {
  "feedbackCount": "Integer",
  "averageFeedbackScore": "Double",
  "feedbackMessages": ["String"]
},
```

```
  "reportGeneratedDate": "Date"
}
```

Notification

```
{
  "notificationId": "ObjectId",
  "sender": "String",
  "recipients": ["String"],
  "notificationSubject": "String",
  "notificationMessage": "String",
  "notificationStatus": "String",
  "entityNameRelatedToNotification": "String",
  "timeStamp": "Date"
}
```

Team

```
{
  "teamId": "ObjectId",
```

```
  "teamName": "String",
  "teamDescription": "String",
  "teamMembers": ["String"],
  "invitedMembers": ["String"],
  "requestedToJoinMembers": ["String"],
  "teamOwner": "String",
  "creationDate": "Date",
  "teamVisibilityStatus": "String",
  "teamModificationHistories": ["String"],
  "teamChat": [
    {
      "timestamp": "Date",
      "sender": "String",
      "message": "String"
    }
  ],
  "teamTasks": ["String"]
}
```

Feedback

```
{
  "feedbackId": "ObjectId",
  "feedbackSender": "String",
  "feedbackRecipients": ["String"],
  "feedbackTaskName": "String",
  "feedbackScore": "Integer",
  "feedbackMessage": "String",
  "timeStamp": "Date"
}
```

Session

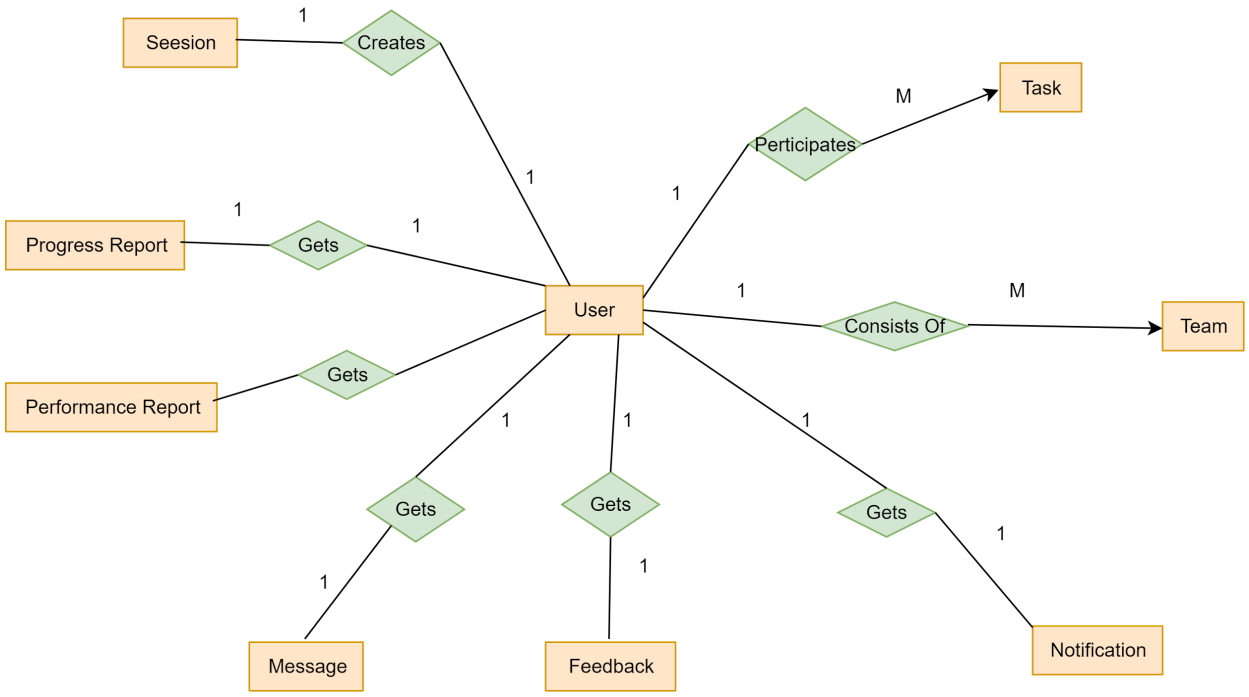
```
{
  "sessionId": "ObjectId",
```

```
  "sessionCreator": "String",
  "sessionTimeStamp": "Date",
  "duration": "Integer",
  "sessionGoal": "String",
  "sessionOutcome": "String",
  "sessionEfficiency": "Integer",
  "tasksOperated": ["String"]
}
```

Message

```
{
  "messageId": "ObjectId",
  "sender": "String",
  "recipients": ["String"],
  "messageSubject": "String",
  "messageDescription": "String",
  "messageStatus": "String",
  "timeStamp": "Date"
}
```

4.3.3 Collection Relationship (CR) Diagram



OBJ OBJ

Figure 11: TimeWise Entity Relationship Diagram

5. Class Based Modeling

5.1 What is Class Based Modeling?

Class-based modeling is a technique used in software development to represent the structure of a system by defining **classes**, their **attributes**, and their **relationships**. A class encapsulates data

(attributes) and behaviors (methods) into a single entity, serving as a blueprint for objects in object-oriented programming. This model helps in visualizing how different parts of a system interact with each other.

5.2 Class Based Modeling: TimeWise

5.2.1 List of Nouns and Related Verbs

No.	Nouns	Related Verbs
01	User	Create, Manage, Update, Delete, Authenticate, Login, Logout, Register, View Profile, Interact, Collaborate
02	Student	Register, Manage Tasks, View Stats, Track Progress, Receive Notifications, Collaborate
03	Teacher	Register, Manage Tasks, View Stats, Assign, Collaborate, Provide Feedback
04	Manager	Assign, Track, Evaluate, Collaborate, Manage Tasks, Review, Provide Feedback
05	Employee	Register, Update, Manage Tasks, Track Progress, Provide Feedback
06	Working Professional	Manage Tasks, Track, View Stats, Collaborate, Receive Notifications
07	Others	Register, Manage Tasks, Track Progress, Collaborate, View Stats
08	Task	Create, Update, Delete, Assign, Track, Prioritize, Share, Complete, Modify, Comment

No.	Nouns	Related Verbs
01	User	Create, Manage, Update, Delete, Authenticate, Login, Logout, Register, View Profile, Interact, Collaborate
02	Student	Register, Manage Tasks, View Stats, Track Progress, Receive Notifications, Collaborate
09	Progress	Track, Record, Evaluate, Update, Visualize
10	Performance	Evaluate, Track, Record, Analyze, Improve
11	Notification	Send, Receive, Track, Read, Acknowledge
13	Stats	Generate, View, Analyze, Interpret
14	RoadMap	Set, Track, Achieve, Edit, Evaluate, Complete
15	Team	Create, Manage, Update, Collaborate, Share
16	Feedback	Provide, Receive, Analyze, Track
17	Feedback Type	Select, Categorize
18	Role	Assign, Change, Manage
19	Category	Assign, Filter, Organize
20	Session	Start, End, Track, Analyze
21	TimeWise	Develop, Deploy, Manage, Use, Maintain
22	LogIn	Authenticate, Validate
23	Register	Register, Create Account
24	Account	Create, Update, Delete, Manage

No.	Nouns	Related Verbs
01	User	Create, Manage, Update, Delete, Authenticate, Login, Logout, Register, View Profile, Interact, Collaborate
02	Student	Register, Manage Tasks, View Stats, Track Progress, Receive Notifications, Collaborate
25	Authentication	verify,check
26	Deadline	Set, Track, Extend, Adjust
27	Priority	Set, Adjust, Change
28	Comments	Add, Edit, Delete, View
29	Timer	Start, Stop, Track, Adjust
30	Productivity Meter	Track, Display, Measure
31	Task Sharing	Share, Collaborate, Modify, View
32	Task logs	Administer, Approve, Modify, Track, Delete
33	Analytics Engine	Analyze, Generate Reports, Track, Process
34	User Dashboard	View, Update, Manage
35	Date and time service	Track, Adjust, Provide
36	Goal tracking	Track, Analyze, Update
37	Progress Analytics	Analyze, Track, Interpret
38	Performance Analytics	Analyze, Track, Visualize
39	Reporting service	Generate, Deliver, Track
40	Performance logs	Record, Store, Track

No.	Nouns	Related Verbs
01	User	Create, Manage, Update, Delete, Authenticate, Login, Logout, Register, View Profile, Interact, Collaborate
02	Student	Register, Manage Tasks, View Stats, Track Progress, Receive Notifications, Collaborate
41	Statistics engine	Process, Analyze, Generate
42	Visualization service	Render, Display, Update
43	Focus timer service	Start, Stop, Track, Adjust
44	Efficiency analytics	Analyze, Track, Improve
45	Session Analytics	Analyze, Track, Summarize
46	Task status	Update, View, Track
47	Collaboration engine	Facilitate, Manage, Coordinate
48	Central database	Store, Retrieve, Update, Manage
49	Task completion stats	Track, Record, Display
50	Time spent stats	Track, Analyze, Summarize
51	Category based stats	Generate, Track, Display
52	Progress reports	Generate, View, Track
53	System Services	Provide, Integrate, Manage

5.2.2 General Classifications

Candidate classes are categorized based on the seven general classification. The analysis classes manifest themselves in one of the following ways:

1. External entities
2. Things
3. Events
4. Roles
5. Organizational units
6. Places
7. Structures

A candidate class is selected for special classification if it fulfills three or more characteristics

No.	**Nouns**	**General Classification**
01	User	1, 4, 5
02	Student	1, 4, 5
03	Teacher	1, 4, 5
04	Manager	1, 4, 5
05	Employee	1, 4, 5
06	Working Professional	1, 4, 5
07	Others	1, 4
08	Task	2, 3, 4
09	Progress	3
10	Performance	3

No.	Nouns	General Classification
01	User	1, 4, 5
02	Student	1, 4, 5
11	Notification	3
12	Reminder	3
13	Stats	3
14	Goal	3
15	Team	1, 4, 5
16	Feedback	3, 5
17	Feedback Type	4
18	Role	4
19	Category	2
20	Session	3
21	TimeWise	1, 2, 5
22	LogIn	3
23	Register	3
24	Account	2
25	Authentication	2,3,5
26	Deadline	3
27	Priority	3

No.	Nouns	General Classification
01	User	1, 4, 5
02	Student	1, 4, 5
28	Comments	3
29	Timer	3
30	Productivity Meter	3, 6
31	Task Sharing	2, 3
32	Task Logs	1, 4, 5
33	Analytics Engine	1, 2, 7
34	User Dashboard	2, 5, 7
35	Date and time service	2, 3, 5
36	Goal tracking	2, 3
37	Progress Analytics	2, 3, 5
38	Performance Analytics	2, 3, 5
39	Reporting service	2, 3, 5
40	Performance Report	3
41	Statistics engine	2, 5, 7
42	Visualization service	2, 5
43	Focus timer service	2, 3
44	Efficiency analytics	2, 3, 5

No.	Nouns	General Classification
01	User	1, 4, 5
02	Student	1, 4, 5
45	Session Analytics	2, 3, 5
46	Task status	2, 3
47	Collaboration engine	1, 2, 5
48	Central database	2, 5, 7
49	Task completion stats	3, 6
50	Time spent stats	3, 6
51	Category based stats	3, 5
52	Progress reports	2, 3, 5
53	System Services	1, 2, 5

5.2.3 Selection Criteria

The candidate classes are then selected as classes by six Selection Criteria:

1. Retain information
2. Needed services
3. Multiple attributes
4. Common attributes
5. Common operations
6. Essential requirements

A candidate class generally becomes a class when it fulfills around three characteristics.

No	Potential Classes	Selection Criteria
01	User(Student,Teacher,Working Professional,Employee,Managers,Others)	1, 3, 5, 6
02	Task	1, 2, 3, 5
03	Progress	1, 3, 5
04	Performance	1, 2, 3, 5, 6
05	Authentication	2, 5, 6
06	Roadmap	1, 2, 3, 5
07	Team	1, 2, 5, 6
08	Feedback	1, 3, 6
09	Session	1, 2, 5
10	Account	1, 5
11	Analytics Engine (Session Analytics, Progress Analytics, Performance Analytics,Efficiency Analytics)	1, 2, 3, 5
12	User Dashboard	2
13	System Services (Date Time Service, Reporting Service, Visualization Service)	2, 5, 6

14	Progress Report	1, 5
15	Statistics Engine	1, 2, 3, 5
16	Collaboration Engine	1, 2, 3, 5
17	Focus Timer Service	1, 2, 6
18	Performance Report	1, 5
19	Task Logs	1, 2, 6
20	Task Completion Stats	4, 5
21	Time Spent Stats	4, 5
22	Category Based Stats	4, 5
23	Performance Logs	4, 5
24	Comments	4, 5
25	Admin	1, 3, 5, 6
26	Productivity Meter	4, 5
27	Timer	4, 5
28	Notification	1, 3, 5

5.2.4 Selected Classes

These are main and central classes in this system:

Selected Classes
User (Student,Teacher,Manager,Employee,Working Professional,Other)
Task
Progress Report
Performance Report
Notification
Team
Feedback
Session
Analytics Engine (Session Analytics, Progress Analytics, Performance Analytics,Efficiency Analytics)
System Services (Date Time Service, Reporting Service, Visualization Service)
Collaboration Engine
Statistics Engine

5.3 Class-Card

1. User

- **Attributes**:
    - **userId** (ObjectId) - Unique identifier for the user.
    - **userName** (String) - Name of the user.
    - **userEmail** (String) - Email of the user.
    - **password** (String) - Password for the user's account.
    - **shortBioData** (String) - Short biography or description of the user.
    - **role** (String) - Role of the user (e.g., student, teacher).
    - **userStatus** (String) - Active or inactive status of the user.
    - **accountVisibility**(String) - Account can set visibility
    - **usersFollowing** (Set of Strings) - Users being followed by this user.

- **Methods**:
    - **initiateRegistration()**
    - **completeRegistration()**
    - **userLogin()**
    - **forgotUserCredentials()**
    - **forgottenAccountVerification()**
    - **getUserDetails()**
    - **getUsersPersonalAccountDetails()**
    - **updateUserAccountDetails()**
    - **getAllNotifications()**
    - **getSearchResult()**
    - **followOrUnfollowUser()**

2. Task

- **Attributes**:
    - **taskId (ObjectId)** - Unique identifier for the task.
    - **taskName (String)** - Name of the task.
    - **taskCategory (String)** - Category of the task.
    - **taskDescription (String)** - Description of the task.
    - **taskPriority (String)** - Priority level of the task.
    - **taskVisibilityStatus (String)** - Public or private visibility of the task.
    - **taskCreationDate (Date)** - Date the task was created.
    - **taskDeadline (Date)** - Deadline for task completion.
    - **taskOwner (String)** - Owner of the task.
    - **taskGoal (String)** - Goal of the task.
    - **taskParticipants (Set of Strings)** - Participants involved in the task.
    - **invitedMembers (Set of Strings)** - Members invited to join the task.
    - **taskComments (List of Comment)** - Comments related to the task.
    - **taskNotes (TreeMap<String, List<Note>>)** - Notes organized by users.
    - **taskCurrentProgress (Integer)** - Progress percentage of the task.
    - **taskModificationHistory (List of TaskModification)** - History of modifications made to the task.
    - **taskTodos (List of TaskTodo)** - To-do items within the task.

- **Methods**:
    - **createTask()**
    - **addTaskComment()**
    - **deleteTaskComment()**
    - **addTaskNote()**
    - **deleteTaskNote()**
    - **addTaskTodo()**
    - **`deleteTaskTodo()`**
    - **`modifyTaskAttribute()`**
    - **`updateTaskTodoStatus()`**

- `calculateTaskProgress()`
- `deleteTask()`
- `getAllTasksSummaryOfAnUser()`
- `getTaskProfile()`
- `getTaskDetails()`
- `inviteMembers()`
- `handleInvitationResponse()`

3. Progress Report

- **Attributes**:
  - **progressReportId (ObjectId)** - Unique identifier for the progress report.
  - **userName (String)** - Name of the user associated with the progress report.
  - **taskStatuses (List<TaskStatus>)** - List of task statuses in the report.
  - **timeStamp (Date)** - Timestamp when the progress report was created.

****TaskStatus (Inner Class of ProgressReport)**

1. **taskName (String)** - Name of the task.
2. **taskOwner (String)** - Owner of the task.
3. **taskPriority (String)** - Priority level of the task.

4. **tasksCurrentProgress (Integer)** - Current progress percentage of the task.
5. **taskCreationDate (Date)** - Creation date of the task.
6. **taskDeadline (Date)** - Deadline for the task.
7. **isDeadlineCrossed (Boolean)** - Indicator if the deadline has been crossed.

- **Methods:**

  addTaskStatuses(Object Task)

  getCurrentProgressReport()

  getPreviousProgressReports()

4. Performance Report

- **Attributes**:
  - **performanceReportId (ObjectId)** - Unique identifier for the performance report.
  - **userName (String)** - Username of the user for whom the report is generated.
  - **usersAccountStatistics (UsersAccountStatistics)** - Account-related statistics of the user.

- **usersTaskStatistics (UsersTaskStatistics)** - Task-related performance statistics of the user.
- **usersSessionStatistics (UsersSessionStatistics)** - Session-based productivity statistics of the user.
- **usersFeedbackStatistics (UsersFeedbackStatistics)** - Feedback and review statistics related to the user.
- **reportGeneratedDate (Date)** - Date when the performance report was generated.

Method:

1)getPerformanceReport()

2)getPreviousPerformanceReports()

5. Notification

- Attributes:
  - **`notificationId (ObjectId)`** - `Unique identifier for the notification.`
  - **`sender (String)`** - `Username of the sender.`

- **recipients (Set of Strings)** - Usernames of the recipients.
- **notificationSubject (String)** - Subject of the notification (can be an enum).
- **notificationMessage (String)** - Content of the notification.
- **notificationStatus (String)** - Status of the notification (e.g., Seen, Unseen).
- **entityNameRelatedToNotification (String)** - Name of the entity related to the notification (e.g., team-related notifications).
- **timeStamp (Date)** - Timestamp of when the notification was created.

6.Message

- **Attributes**:

  messageId (ObjectId) - Unique identifier for the message.

  sender (String) - Username of the sender.

  recipients (Set of Strings) - Usernames of the recipients.

  messageSubject (String) - Subject of the message.

  messageDescription (String) - Content or body of the message.

  messageStatus (String) - Status of the message (e.g., Read, Unread).

  timeStamp (Date) - Timestamp indicating when the message was sent.

Methods:

`sendMessage()`

`getAllMessages()`

7. Team

- **Attributes:**
  - **createTeam()**
  - **getTeamsForUser()**
  - **getTeamProfile()**
  - **getTeamDetails()**
  - **addTaskToTeam()**
  - **removeTaskFromTeam()**
  - **removeMemberFromTeam()**
  - **leaveTeam()**
  - **addTeamChat()**
  - **removeTeamChat()**
  - **inviteMembers()**
  - **handleInvitationResponse()**
  - **updateTeamDetails()**
  - **requestTeamJoining()**
  - **handleTeamJoiningRequestResponse()**

- **Methods:**

inviteMembersToTeam()

createTeam()

addTaskToTeam()

removeTaskFromTeam()

removeUserFromTeam()

leaveTeam()

getTeamDetails()

getTeamsForUser()

8. Feedback

- **Attributes:**
  - **feedbackId (ObjectId)** - Unique identifier for the feedback.
  - **feedbackSender (String)** - Username of the user who provided the feedback.
  - **feedbackRecipient (String)** - Username of the user receiving the feedback.
  - **feedbackTaskName (String)** - Name of the task associated with the feedback.
  - **feedbackScore (Integer)** - Score given in the feedback (range: 0 to 100).

- **feedbackMessage (String)** - Detailed message or comments in the feedback.
- **timeStamp (Date)** - Timestamp indicating when the feedback was given.

- **Methods:**
  - `sendFeedback()`
  - `getAllFeedbacks()`
  - `removeFeedback()`

10. Session

- **Attributes**:
  - **sessionId (ObjectId)** - Unique identifier for the session.
  - **sessionCreator (String)** - Username of the user who created the session.
  - **sessionTimeStamp (Date)** - Timestamp indicating when the session was created.
  - **duration (Integer)** - Duration of the session in minutes.
  - **sessionGoal (String)** - Goal or objective of the session.
  - **sessionOutcome (String)** - Outcome or result of the session.
  - **sessionEfficiency (Integer)** - Efficiency rating of the session (e.g., percentage).
  - **tasksOperated (Set of Strings)** - Tasks worked on during the session.
  -

- **Methods**:
    - `createSession()`
    - `getSessionById()`
    - `deleteSession()`
    - `getAllSessionsOfAnUser()`

Engines:

Analytics Engine :
Methods:

`initStaticDependencies()`
`generateWeeklyPerformanceReport()`
`generatePerformanceReport()`
`generateProgressReport()`
`getProgressReportsByUser()`
`deleteProgressReport()`
`getUsersDeepWorkingHours()`
`calculateHourDistribution()`
`getHour()`
`getTimeSlot()`

Statistics Engine:
Methods:

```
initStaticDependencies()
calculateTaskStatistics(String userName, int numberOfDays)
calculateSessionStatistics(String userName, int numberOfDays)
calculateFeedbackStatistics(String userName, int numberOfDays)
calculateUserAccountStatistics(String userName,int numberOfDays)
calculateCutoffDate(int numberOfDays)
```

Collaboration Engine:
Methods:

```
initStaticDependencies()
sendMessage()
sendFeedback()
createNotification()
createTeamNotification()
handleTeamJoiningInvitation()
handleTaskParticipatingInvitation()
handleTeamJoiningInvitationResponse()
handleTaskParticipatingInvitationResponse()
sendMailFromTimeWise()
sendEmail()
sendProgressAndPerformanceReport()
sendUserRegistrationVerificationCode()
sendRegistrationSuccessfulMessage()
sendForgottenAccountCredentials()
sendAccountVerificationCode()
addTaskToTeam()
removeTaskFromTeam()
removeDeletedTaskFromTeams()
```

removeMemberFromTeam()
leaveTeam()
handleTeamJoiningRequest()
handleTeamJoiningRequestResponse()
getAllNotificationsForUser()

System Service:
Method:
initStaticDependencies()
checkRegistrationCredentialsAndSendRegistrationVerificationCode(User user)
verifyVerificationCodeAndCompleteRegistration(User user, String code)
performUserLogin(LoginCredentials loginCredentials)
handleForgotUserCredentials(String userEmail)
verifyVerificationCodeForAccountVerification(String code, String userEmail, String userName, String updatedPassword)
generateVerificationCode(User user)
updateUserAccountDetails(String userName, UpdatedUserAccount updatedUserDetails)
getUserAccountDetails(String currentUserName, String userName)
sendDailyProgressReportToAllUsers()
sendWeeklyPerformanceReportToAllUsers()
processReports(Set<String> userNames, Consumer<String> reportProcessor)
generateAndSendDailyProgressReport(String userName)
generateAndSendWeeklyPerformanceReport(String userName)
logError()

```
generateMessageDescriptionFromProgressReport()
generateMessageDescriptionFromPerformanceReport()
getSearchResult()
matchesKeyword()
calculateMatchScore()
```

5.4 Class Responsibility(CRC) Diagram

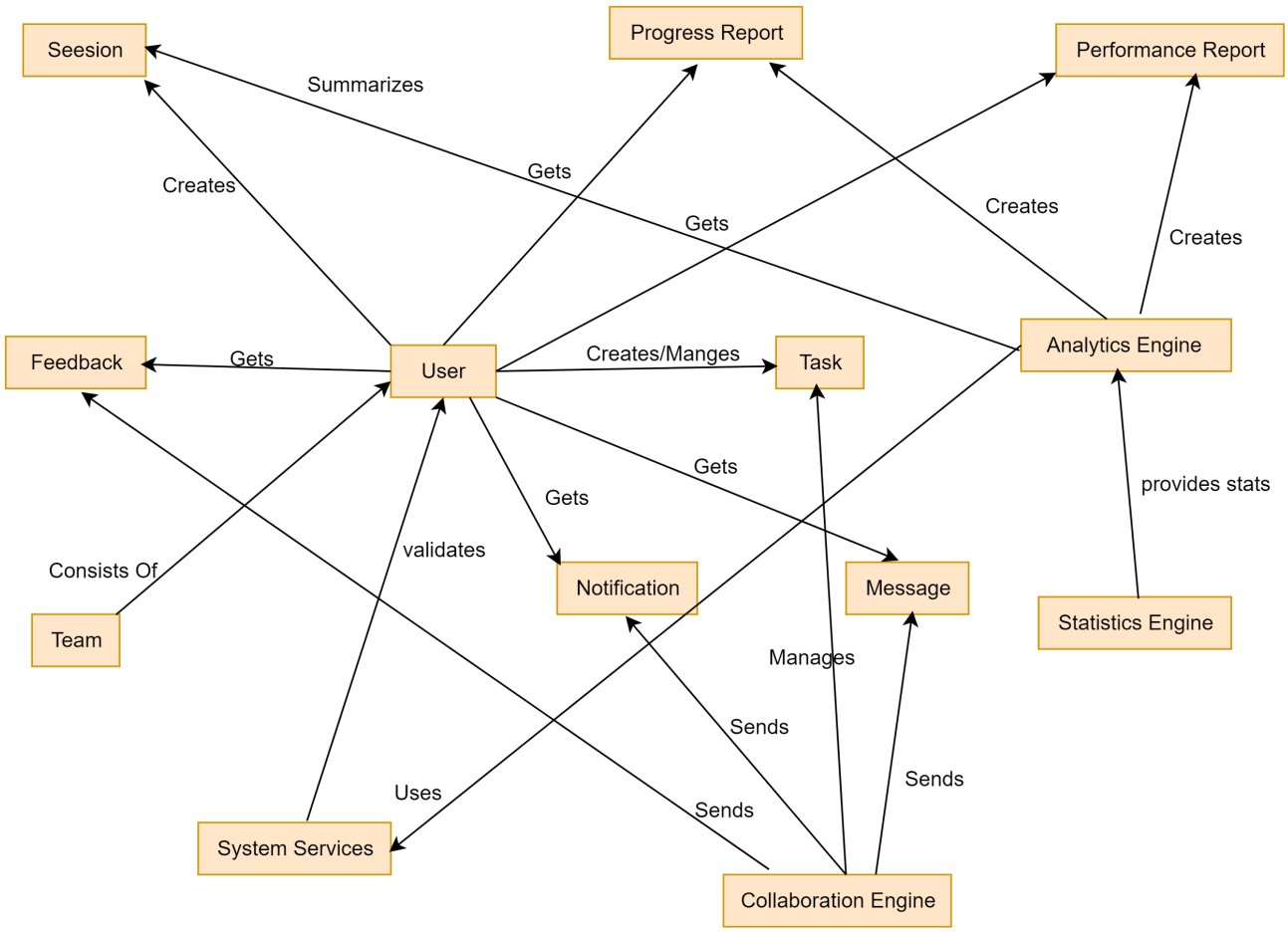


Figure 12: TimeWise CRC Diagram

7. Behavioral Modeling

The behavioral model indicates how the software will respond to external events or stimuli. In the context of behavioral modeling, two different characterizations of states must be considered: (1) the state of each class as the system performs its function and (2) the state of the system as observed from the outside as the system performs its function.
The behavioral modeling of TimeWise provides insights into how the system responds to external events, emphasizing both the state of individual classes and the overall system state. Below is a breakdown of the UML State Transition Diagram and a detailed list of events, focusing on the interactions between users, admins, and system components.

Based on the document you've provided, I'll update the State Transition Diagram events while maintaining the same headings but modifying the contents to reflect the TimeWise system described in your document.

7.1 State Transition Diagram for TimeWise

The State Transition Diagram represents the active states of core classes in TimeWise and the events (triggers) that cause transitions. The states are represented as rectangles, and transitions are represented by arrows with labels for the corresponding events.

7.1.1 List of Events

1. User Events

- Registers in the system with name, email, and role
- Logs into the system using username and password
- Recovers account through email verification

- Updates profile information and visibility status
- Follows other users and views their profiles
- Creates tasks with descriptions, priorities, and deadlines
- Comments on tasks and takes personal notes
- Marks todos as complete to update task progress
- Views task modification history
- Receives and manages notifications
- Views progress and performance reports
- Sends messages and feedback to other users
- Creates and manages working sessions
- Views deep work analytics
- Creates teams and manages team membership

2. Task Events

- Creates task with required attributes (name, description, priority, deadline)
- Adds todos to tasks
- Shares tasks with other users
- Updates task progress based on todo completion
- Marks task as complete when all todos are done
- Logs changes to task details and progress
- Gets filtered and sorted based on different criteria
- Receives comments and notes from users

4. Progress Report Events

- Generates daily progress reports for unfinished tasks
- Updates based on task completion status
- Shows deadline and current progress information
- Gets filtered by user-defined time periods
- Displays visualizations of progress data

5. Performance Report Events

- Generates weekly performance reports

- Collects statistical data on tasks, feedback, and account activity
- Creates visualizations (bar graphs, pie charts) of performance metrics
- Gets filtered by user-defined previous number of days
- Shows insights for productivity improvement

6. Notification Events

- Alerts users about task invitations and assignments
- Notifies about team invitations and changes
- Sends verification codes for registration and account recovery
- Gets sorted and filtered by different categories
- Can be removed from the notification panel by users

7. Team Events

- Creates team with unique name
- Adds team members through invitations
- Accepts requests to join team
- Updates team details (owner only)
- Assigns and removes tasks within the team
- Logs team history of member additions, removals, and detail changes
- Provides team chat functionality

8. Feedback Events

- Creates feedback with task name and feedback score
- Sends feedback to other users via email
- Gets stored and managed in feedback dashboard
- Can be removed from the dashboard
- Contributes to user performance metrics

9. Goal Events

- Sets task goals for broader task categorization
- Defines session goals for working sessions
- Generates roadmaps based on goals using AI

- Tracks goal completion through associated tasks

11. Session Events

- Creates session with defined goal, duration, and preferred environment
- Starts and manages stopwatch for session timing
- Logs operated tasks during session
- Saves session notes as outcomes
- Preserves session state when interrupted
- Records session efficiency score
- Stores session history for future reference
- Contributes data to deep work analytics

14. Analytics Engine Events

- Analyzes time spent on sessions
- Calculates productivity scores for different hours of the day
- Creates productivity meter based on session efficiency
- Generates visualizations of deep work patterns
- Provides insights based on session data
- Creates personalized results based on user-defined time periods

15. System Services Events

- Sends daily and weekly reports automatically
- Validates user inputs during registration
- Maintains unique username requirements
- Tracks deadlines for tasks
- Filters and sorts data based on user preferences
- Generates visualizations for statistics

16. Collaboration Engine Events

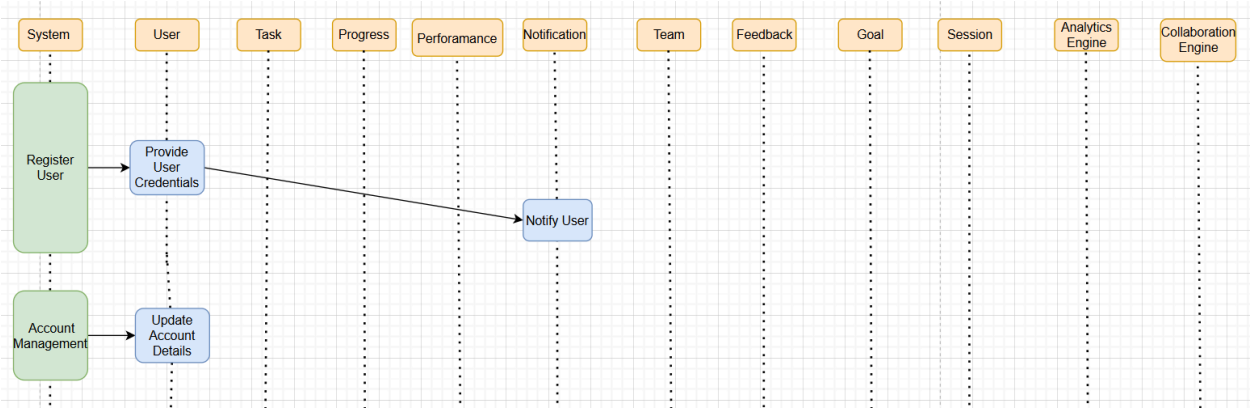
- Shares tasks between users
- Creates collaborative environments in teams
- Manages team chat functionality

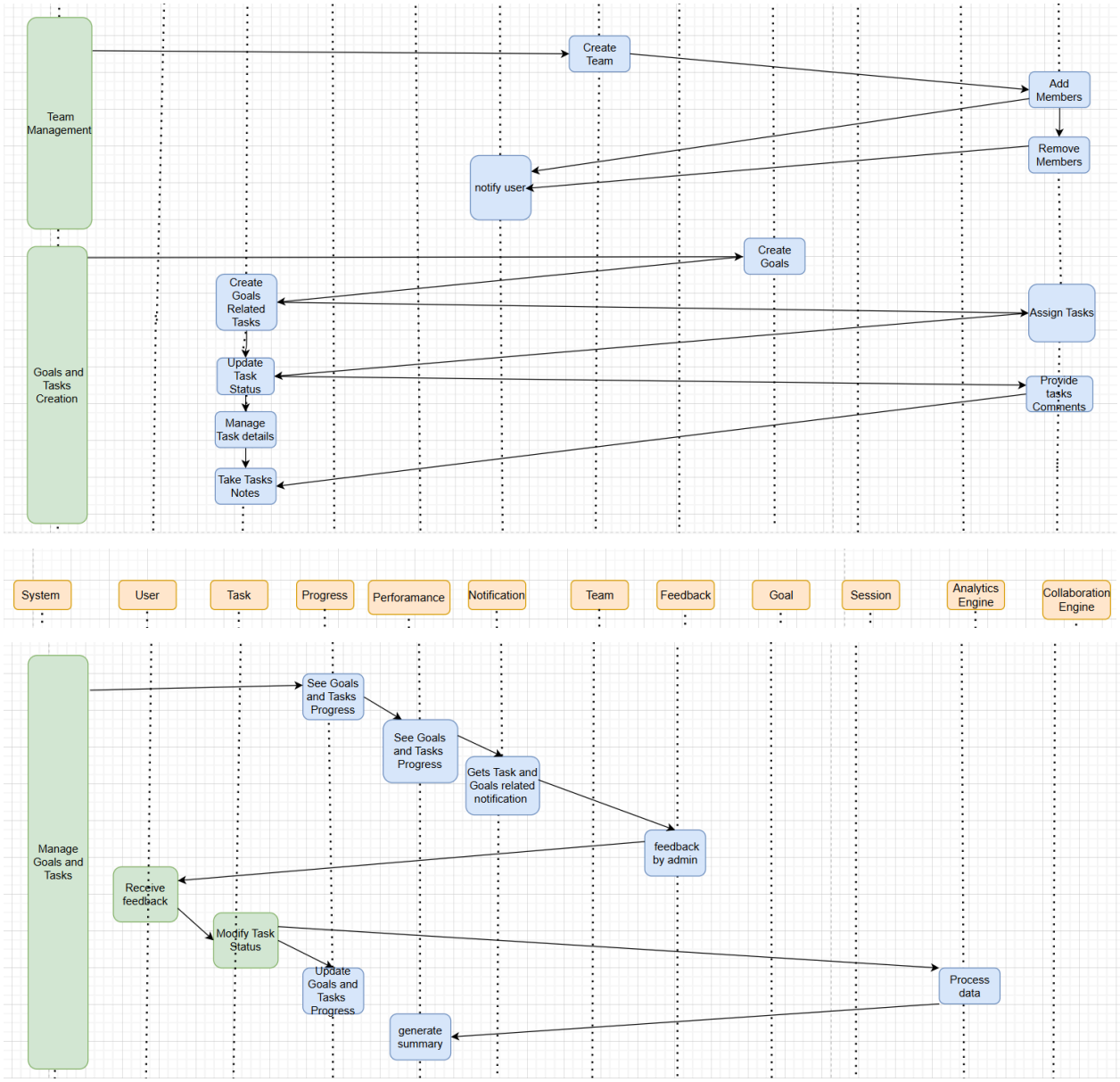
- Handles task participation changes
- Updates access rights when team membership changes
- Notifies members of collaborative events

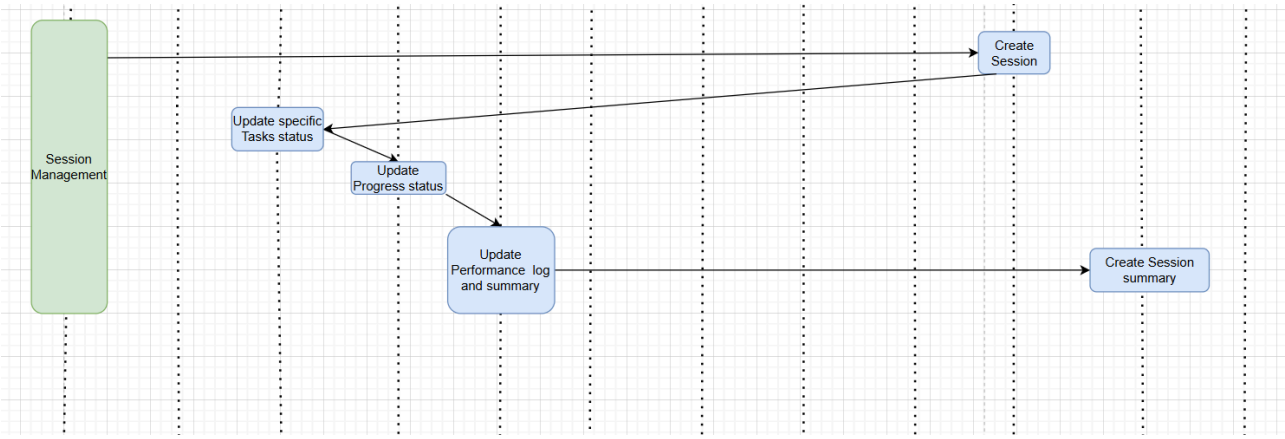
17. Statistics Engine Events

- Generates visualizations for task statistics
- Creates charts for session statistics
- Provides feedback statistics
- Shows account-related statistics
- Filters statistics by previous number of days
- Creates multiple visualization types for better understanding

7.2 Sequence Diagram:

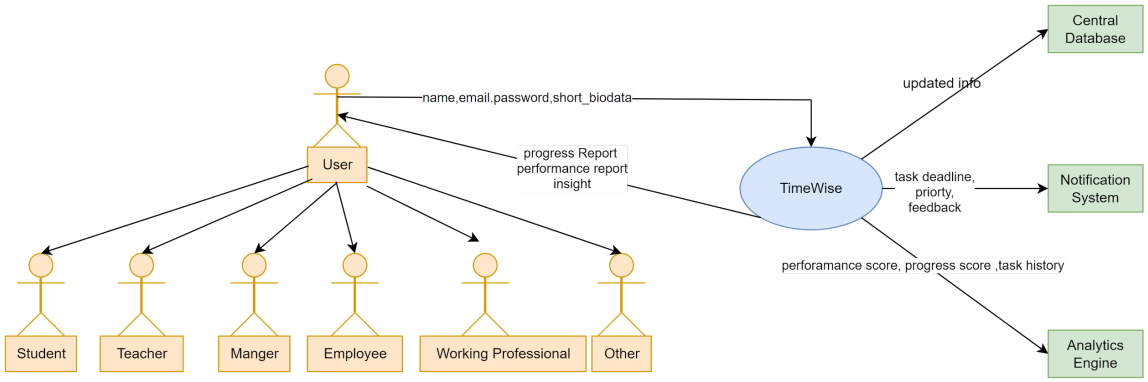



Team Management
Create Team
Add Members
Remove Members
notify user
Goals and Tasks Creation
Create Goals
Create Goals Related Tasks
Assign Tasks
Update Task Status
Provide tasks Comments
Manage Task details
Take Tasks Notes
System
User
Task
Progress
Perforamance
Notification
Team
Feedback
Goal
Session
Analytics Engine
Collaboration Engine
Manage Goals and Tasks
See Goals and Tasks Progress
See Goals and Tasks Progress
Gets Task and Goals related notification
feedback by admin
Receive feedback
Modify Task Status
Update Goals and Tasks Progress
Process data
generate summary

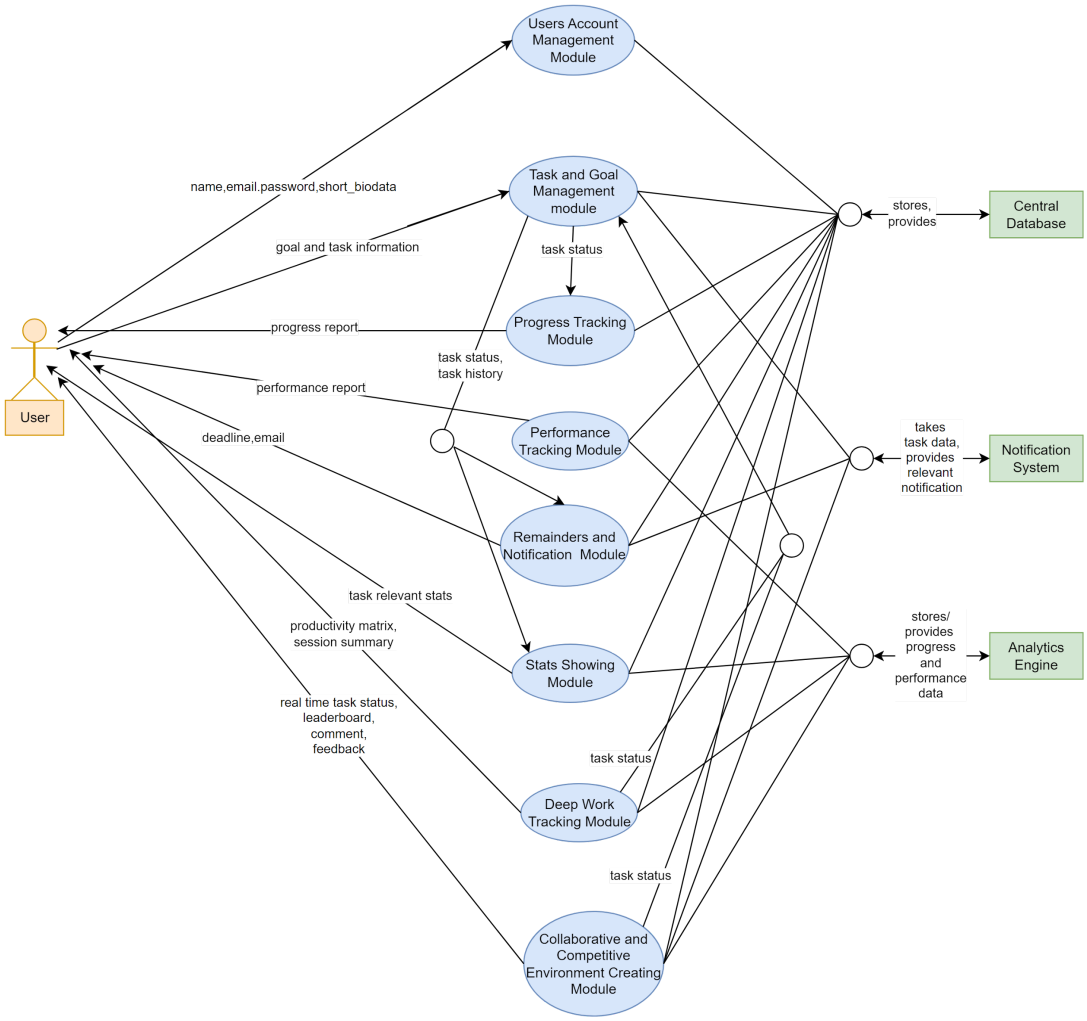


8. DFD diagram:

Level 0:


Central Database
updated info
name,email.password,short_biodata
User
progress Report
performance report
insight
TimeWise
task deadline,
priorty,
feedback
Notification System
perforamance score, progress score ,task history
Analytics Engine
Student
Teacher
Manger
Employee
Working Professional
Other

Level 1:


Users Account Management Module
Task and Goal Management module
Progress Tracking Module
Performance Tracking Module
Remainders and Notification Module
Stats Showing Module
Deep Work Tracking Module
Collaborative and Competitive Environment Creating Module
User
name,email,password,short_biodata
goal and task information
progress report
performance report
deadline,email
task relevant stats
productivity matrix, session summary
real time task status, leaderboard, comment, feedback
task status
task status, task history
task status
task status
stores, provides
Central Database
takes task data, provides relevant notification
Notification System
stores/ provides progress and performance data
Analytics Engine

Background of the Project:

For implementing the project we did prior study about use cases and probable solutions related to task management and productivity as a whole. Different technologies and cloud services have been used in making of the project. Technologies and some of the techniques that has been used in making the project is discussed below:

Techniques:

Rest API:
The whole Project is made by using the rest api concepts. Where the backend can run and operate independently of the frontend. Some urls and routes have been defined with which the frontend Interact and access data and services.

Model ,View Controller (MVC):
The backend parts of the project is implemented via the Model View Controller design pattern. Where the view part is the whole frontend and completely separate and independent of the backend.

Object Relational Mapping(ORM):
The project uses the ORM techniques where each class is a data object and instead of writing raw query to interact with the data it uses ORM to handle the complexity as well as risk.

Single Page Application(SPA):
Instead of writing raw html,css and javascript the project uses a component library based frontend framework to implement the frontend. The whole frontend code is built into a single page file and by that the user doesn't have to ask for html or frontend field each time reducing the latency.

Library Components:

Instead of writing each individual and reusable part(component) each time for most common use cases the project maintains a list of components to reuse. It also uses some library components to handle some of the visual effects like bar chart, pie chart, analysis graph, different icons, forms,cards, dialog box and so on.

Technologies Used:

Backend: Spring boot
Fronted: Next.Js
Components Libraries: Shad CN Ui, lucid react,zod
Database: MongoDB
AI API: Mistral Ai
Fronted Deployment: Vercel
Backend Deployment: Railway
Database Deployment: Atlas

Key features description of the project:

Unique Names:
The system is implemented in such a way that each username and team name is unique. Each task that a user owns also has to have a unique name. If a user tries to opt for a duplicate name in these things the system will verify the credentials and will reject the request by giving a proper response message.

Autocompleting Content:
The user will be able to draft a message or feedback by autocompleting by giving a proper prompt to provide the context.

Generating Roadmap:
The user will be able to generate a roadmap automatically even if they don't have any prior knowledge about the topic.

Suggestion Of Names:
While selecting the recipient of messages or feedback, inviting a member to a team or task or adding a task to the user, you don't have to remember the username,team name or task name . The system will automatically provide suggestions as a dropdown menu to select based on the users following or participating tasks and teams.

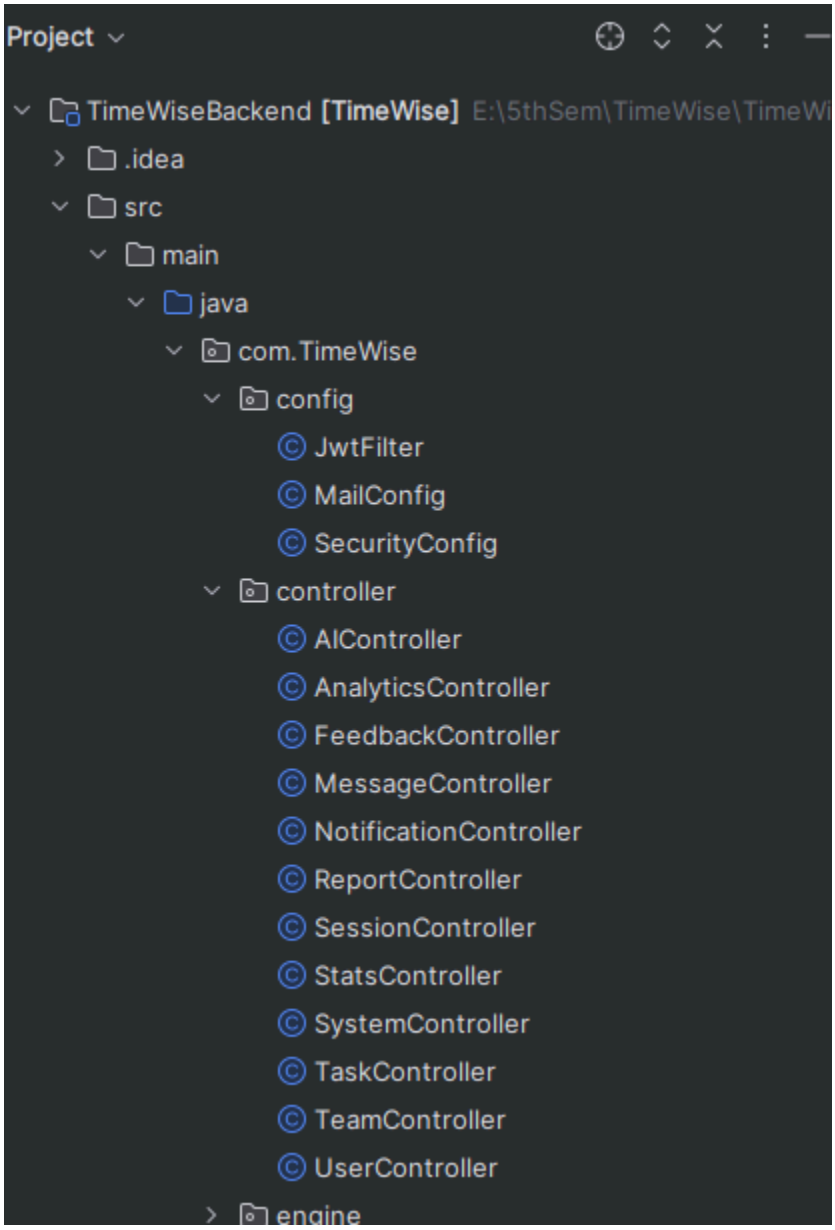
Searching User:
The user can search for other users in the search panel. While searching even if there are some typos the system will handle it and will eventually find and shoe the relevant users based on the user name, user email and user bio.

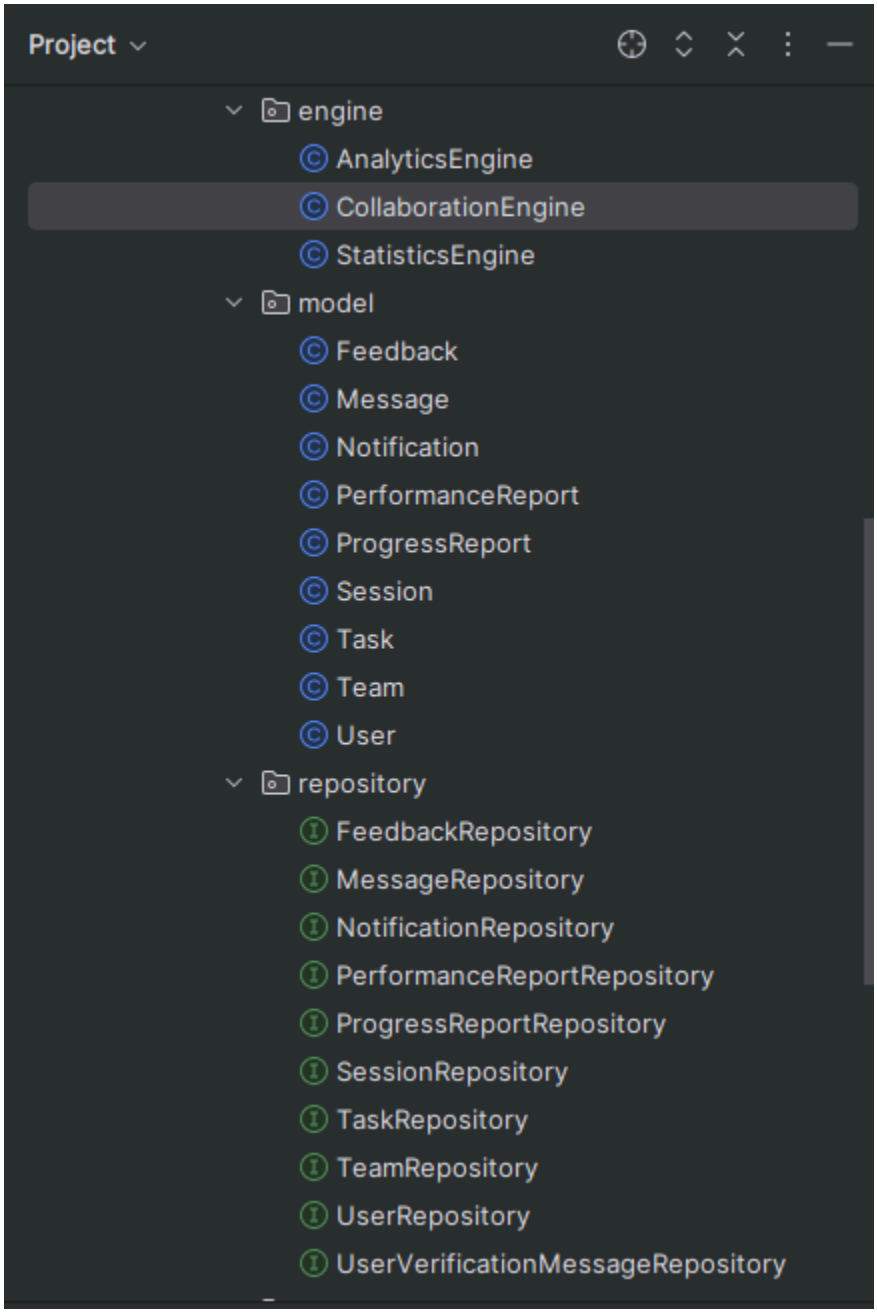
Exploring Profile:
Users can see and explore the other users' account dashboard where they can see what tasks they participate in or what teams they joined.

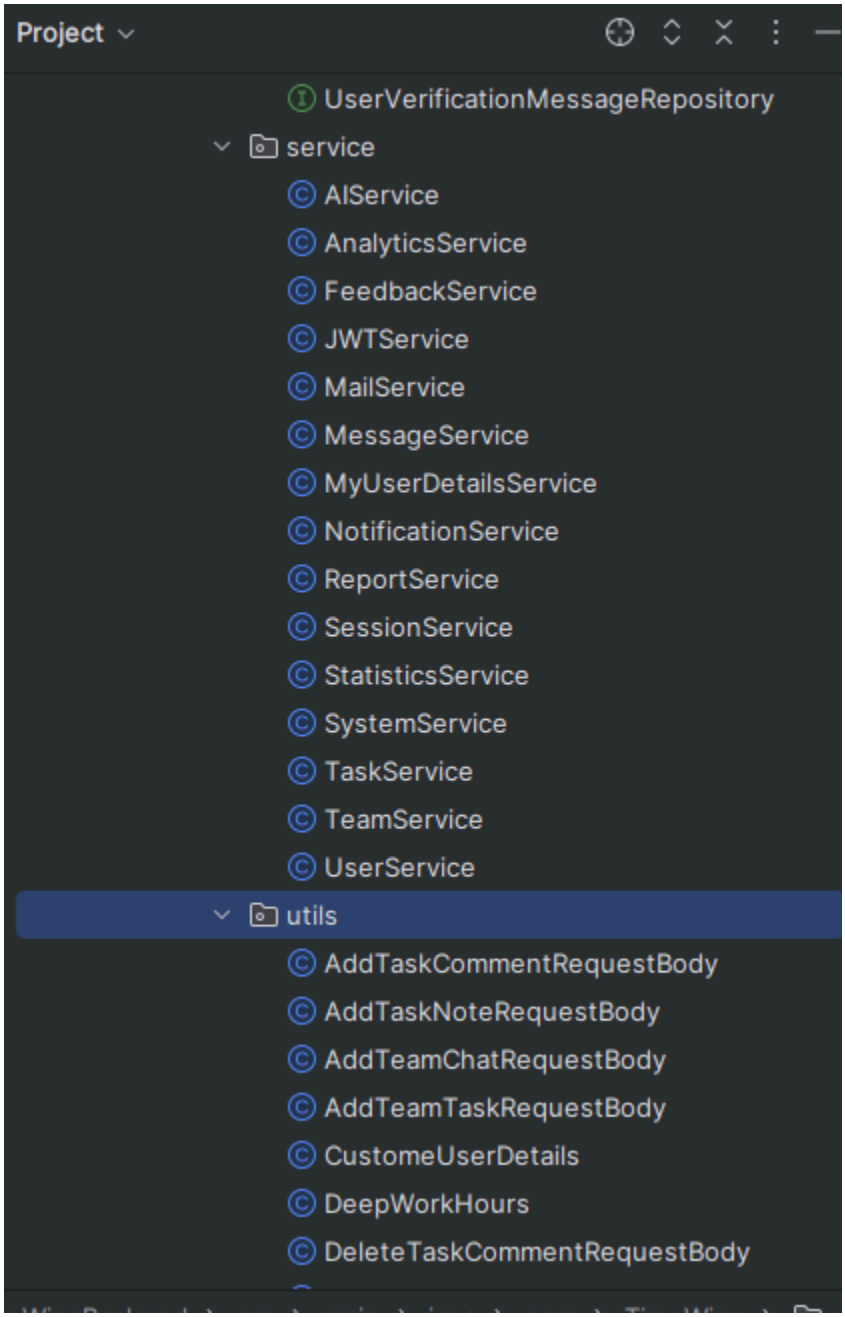
Restricting Access:
If a user sets its account visibility to private then this will not appear in the search results and other users will not be able to explore their profiles. Users can also set individual team and tasks status to private hence these will not appear in the profile dashboard and others will not be able to see it.

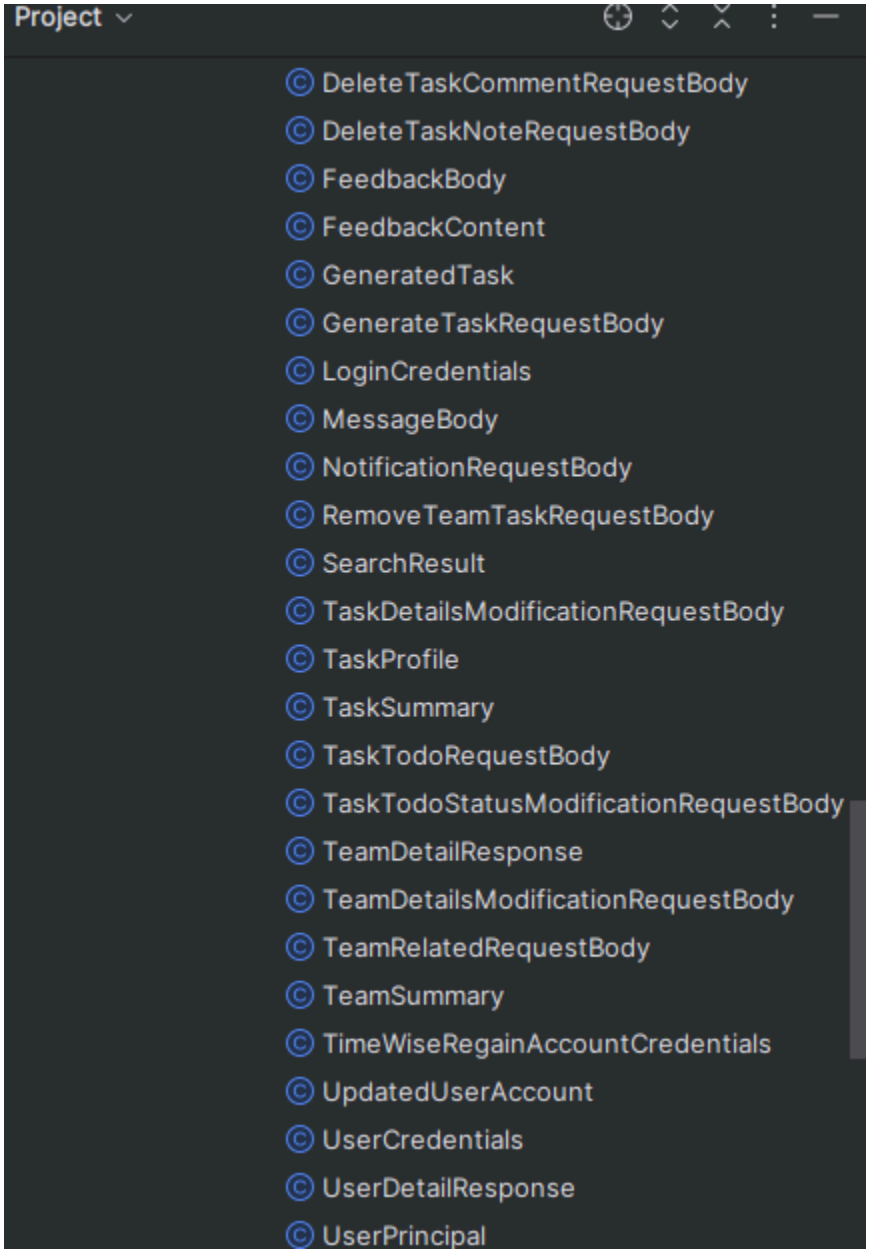
Implementation:

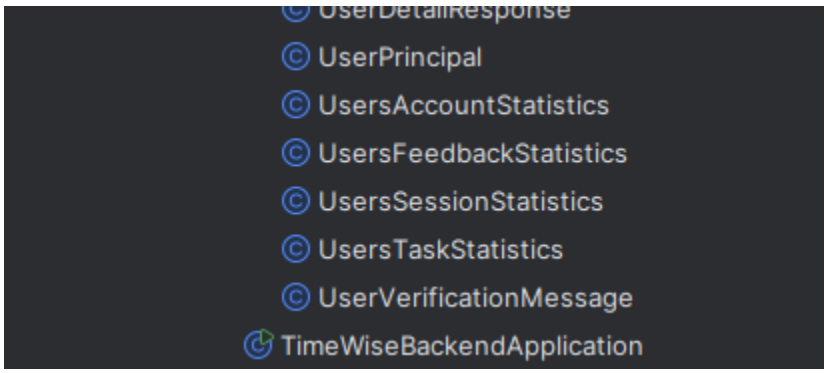
Code Folder Structure:


Project
TimeWiseBackend [TimeWise] E:\5thSem\TimeWise\TimeWi
.idea
src
main
java
com.TimeWise
config
JwtFilter
MailConfig
SecurityConfig
controller
AIController
AnalyticsController
FeedbackController
MessageController
NotificationController
ReportController
SessionController
StatsController
SystemController
TaskController
TeamController
UserController
engine


Project
engine
AnalyticsEngine
CollaborationEngine
StatisticsEngine
model
Feedback
Message
Notification
PerformanceReport
ProgressReport
Session
Task
Team
User
repository
FeedbackRepository
MessageRepository
NotificationRepository
PerformanceReportRepository
ProgressReportRepository
SessionRepository
TaskRepository
TeamRepository
UserRepository
UserVerificationMessageRepository


Project
UserVerificationMessageRepository
service
AIService
AnalyticsService
FeedbackService
JWTService
MailService
MessageService
MyUserDetailsService
NotificationService
ReportService
SessionService
StatisticsService
SystemService
TaskService
TeamService
UserService
utils
AddTaskCommentRequestBody
AddTaskNoteRequestBody
AddTeamChatRequestBody
AddTeamTaskRequestBody
CustomeUserDetails
DeepWorkHours
DeleteTaskCommentRequestBody


Project
DeleteTaskCommentRequestBody
DeleteTaskNoteRequestBody
FeedbackBody
FeedbackContent
GeneratedTask
GenerateTaskRequestBody
LoginCredentials
MessageBody
NotificationRequestBody
RemoveTeamTaskRequestBody
SearchResult
TaskDetailsModificationRequestBody
TaskProfile
TaskSummary
TaskTodoRequestBody
TaskTodoStatusModificationRequestBody
TeamDetailResponse
TeamDetailsModificationRequestBody
TeamRelatedRequestBody
TeamSummary
TimeWiseRegainAccountCredentials
UpdatedUserAccount
UserCredentials
UserDetailResponse
UserPrincipal



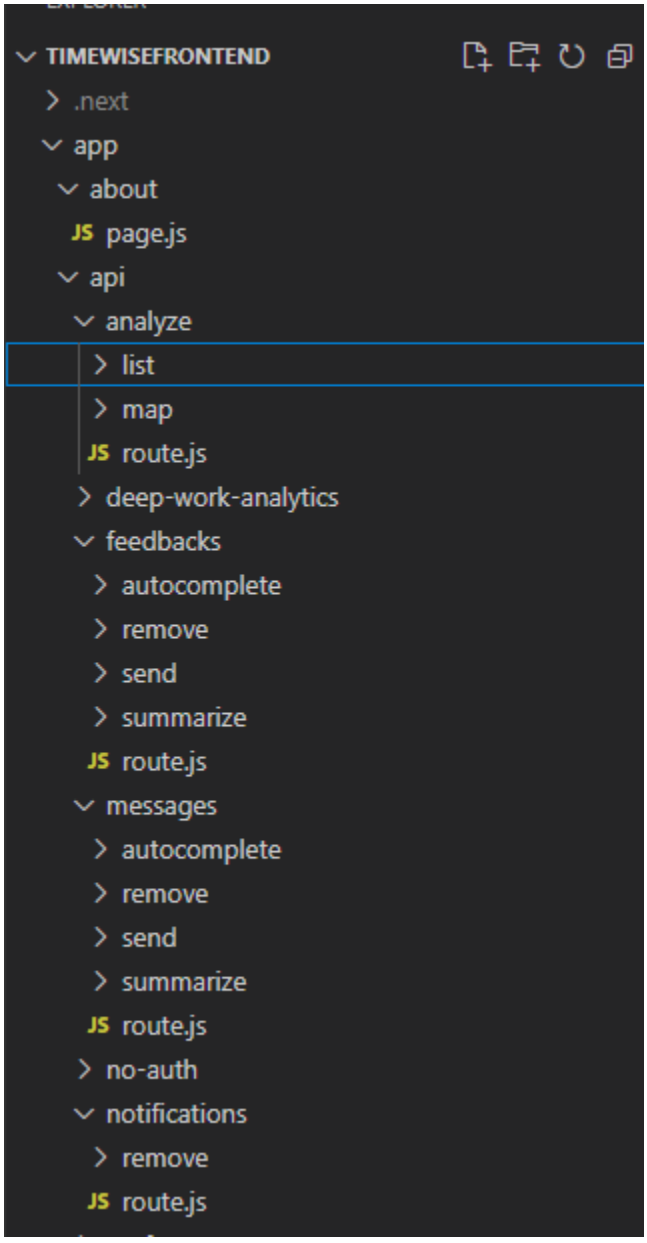
Backend Project Dependencies:

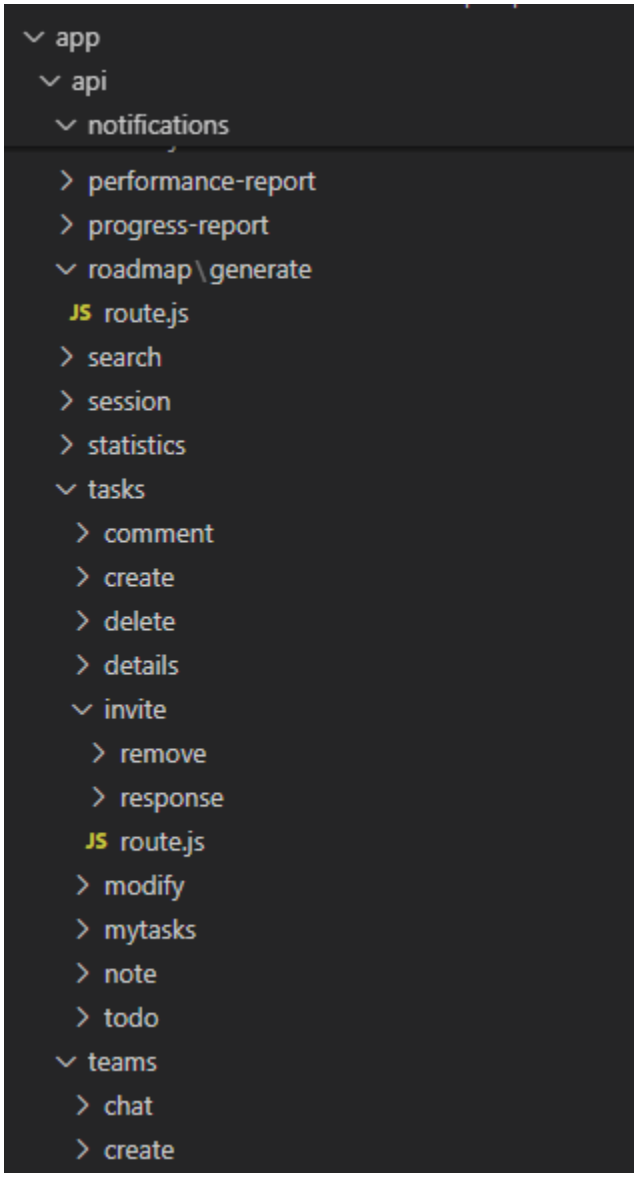
```
<dependencies>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-security</artifactId>
   </dependency>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
   </dependency>
   <!--       <dependency>-->
   <!--          <groupId>org.springframework.boot</groupId>-->
   <!--          <artifactId>spring-boot-starter-data-jpa</artifactId>-->
   <!--       </dependency>-->

   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
   </dependency>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
```

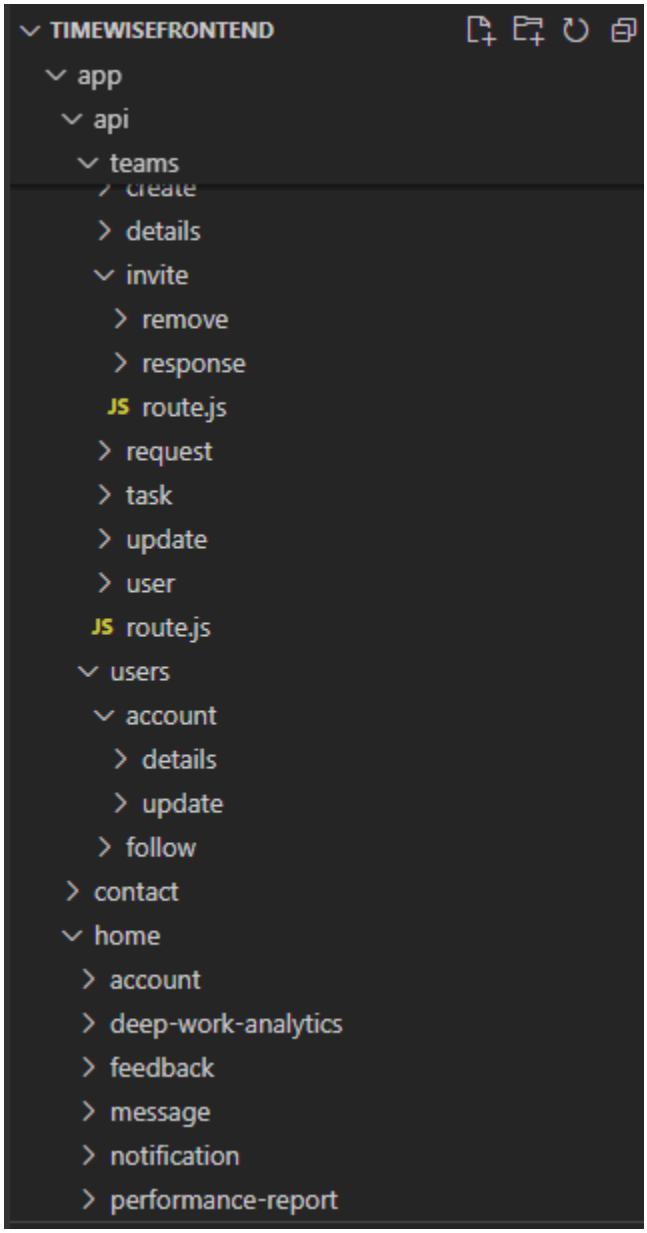
```
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-api</artifactId>
    <version>0.12.5</version>
</dependency>

<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-impl</artifactId>
    <version>0.12.5</version>
    <scope>runtime</scope>
</dependency>

<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-jackson</artifactId>
    <version>0.12.5</version>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
</dependency>

<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
</dependency>  <!-- Spring AI dependency -->
<dependency>
    <groupId>org.springframework.ai</groupId>
    <artifactId>spring-ai-mistral-ai</artifactId>
```

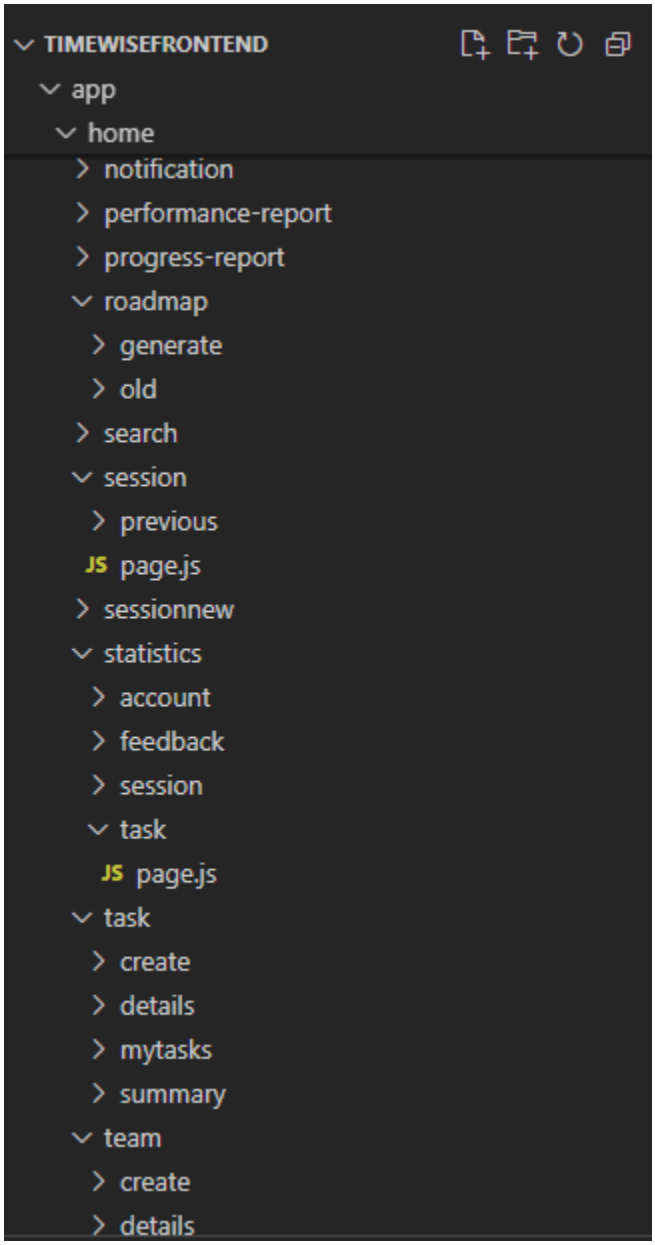
```
    </dependency>

    <dependency>
        <groupId>org.springframework.ai</groupId>
        <artifactId>spring-ai-mistral-ai-spring-boot-starter</artifactId>
        <version>0.8.1</version>
    </dependency>

</dependencies>
```

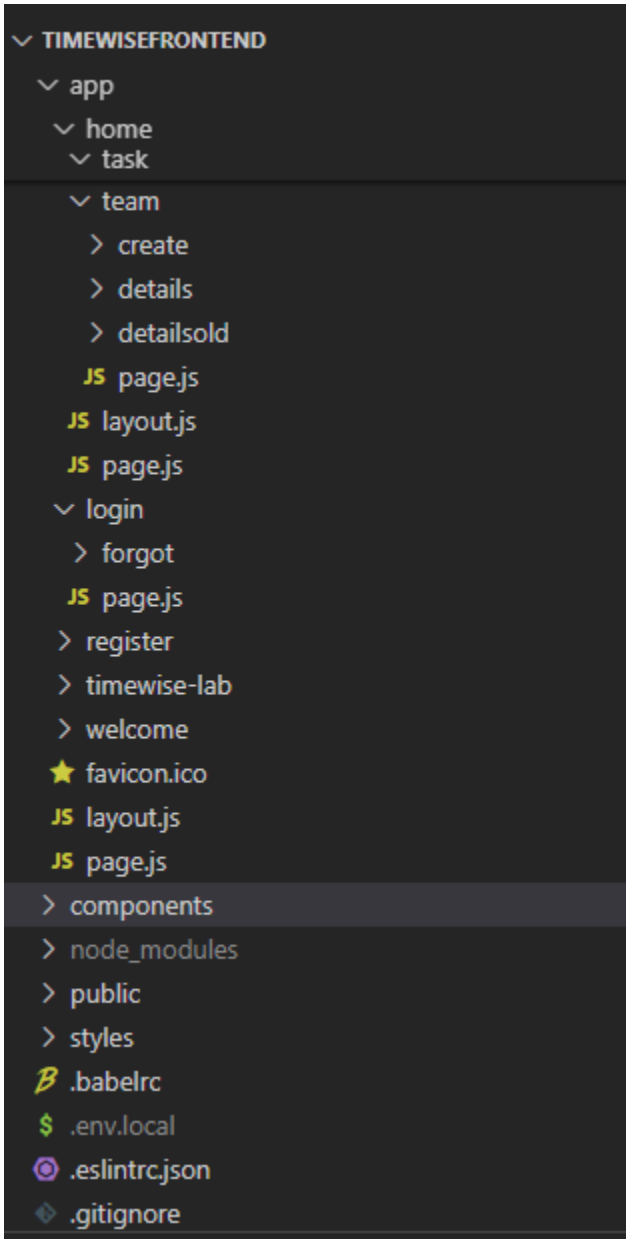
Frontend Folder Structure:


TIMEWISEFRONTEND
.next
app
about
page.js
api
analyze
list
map
route.js
deep-work-analytics
feedbacks
autocomplete
remove
send
summarize
route.js
messages
autocomplete
remove
send
summarize
route.js
no-auth
notifications
remove
route.js


app
api
notifications
performance-report
progress-report
roadmap\generate
route.js
search
session
statistics
tasks
comment
create
delete
details
invite
remove
response
route.js
modify
mytasks
note
todo
teams
chat
create


TIMEWISEFRONTEND
app
api
teams
details
invite
remove
response
route.js
request
task
update
user
route.js
users
account
details
update
follow
contact
home
account
deep-work-analytics
feedback
message
notification
performance-report


TIMEWISEFRONTEND
app
home
notification
performance-report
progress-report
roadmap
generate
old
search
session
previous
page.js
sessionnew
statistics
account
feedback
session
task
page.js
task
create
details
mytasks
summary
team
create
details



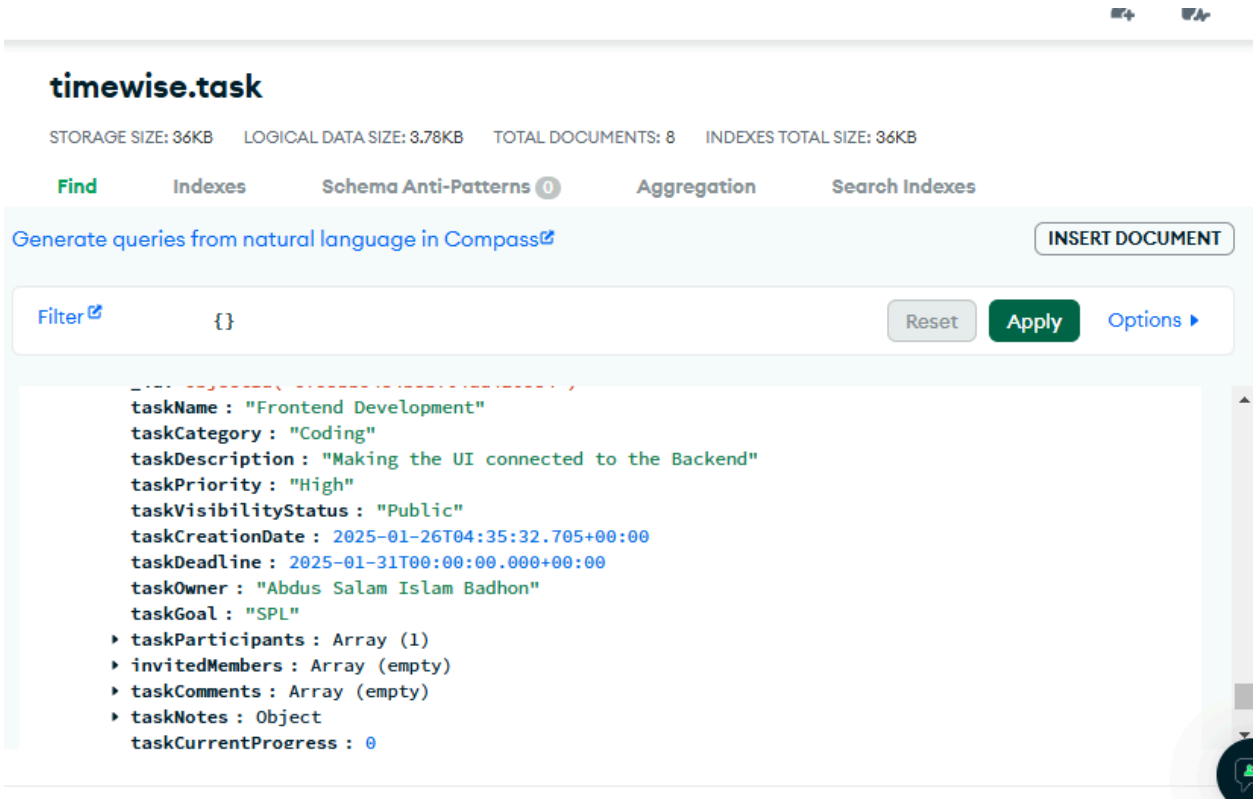
Frontend Dependencies:

```
{
  "name": "timewise",
  "version": "0.1.0",
  "private": true,
  "scripts": {
```

```
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "@babel/runtime": "^7.26.0",
    "@hookform/resolvers": "^3.10.0",
    "@radix-ui/react-accordion": "^1.2.3",
    "@radix-ui/react-alert-dialog": "^1.1.6",
    "@radix-ui/react-arrow": "^1.1.2",
    "@radix-ui/react-avatar": "^1.1.2",
    "@radix-ui/react-checkbox": "^1.1.4",
    "@radix-ui/react-collapsible": "^1.1.3",
    "@radix-ui/react-dialog": "^1.1.5",
    "@radix-ui/react-dropdown-menu": "^2.1.5",
    "@radix-ui/react-label": "^2.1.1",
    "@radix-ui/react-navigation-menu": "^1.2.4",
    "@radix-ui/react-popover": "^1.1.5",
    "@radix-ui/react-primitive": "^2.0.2",
    "@radix-ui/react-progress": "^1.1.1",
    "@radix-ui/react-scroll-area": "^1.2.2",
    "@radix-ui/react-select": "^2.1.5",
    "@radix-ui/react-separator": "^1.1.1",
    "@radix-ui/react-slider": "^1.2.3",
    "@radix-ui/react-slot": "^1.1.2",
    "@radix-ui/react-switch": "^1.1.3",
    "@radix-ui/react-tabs": "^1.1.2",
    "@radix-ui/react-tooltip": "^1.1.7",
    "@shadcn/ui": "^0.0.4",
    "autoprefixer": "^10.4.20",
    "class-variance-authority": "^0.7.1",
    "clsx": "^2.1.1",
    "date-fns": "^3.6.0",
    "framer-motion": "^11.3.24",
    "lucide-react": "^0.474.0",
```

```
    "next": "^15.1.0",
    "next-themes": "^0.4.4",
    "node-fetch": "^3.3.2",
    "react": "^18.0.0",
    "react-day-picker": "^8.10.1",
    "react-dom": "^18.0.0",
    "react-hook-form": "^7.54.2",
    "react-icons": "^5.4.0",
    "react-intersection-observer": "^9.16.0",
    "react-redux": "^9.1.2",
    "react-toastify": "^11.0.5",
    "recharts": "^2.15.1",
    "sonner": "^2.0.1",
    "tailwind-merge": "^2.6.0",
    "tailwindcss-animate": "^1.0.7",
    "zod": "^3.24.1
  },
  "devDependencies": {
    "eslint": "^8",
    "eslint-config-next": "^15.1.0",
    "postcss": "^8.5.1",
    "tailwindcss": "^3.4.17"
  }
}
```

Database:



User Manual:

The following descriptions and figures will serve as a comprehensive guide for all users, providing clear instructions on how to effectively use our system. It covers both initial setup and troubleshooting, ensuring a smooth user experience.

Welcome Page:

Choose the login button to login or register to register within the system. Users can see the different description workings of the different modules in the welcome page.

Registration,Login Account Recovery page:

In the registration page by providing a unique username,valid email address,strong password and optional other information the user can ask for the verification code which will be sent to email. By attaching the verification code

with other user details the user can complete the registration process. For login the user can enter the username and password. If the password or username is forgotten, the user can go to the recovery account page where by entering the recovery email the user will get the verification code and list of usernames associated with the user email in the system. Then users can choose any of the accounts and then option resetting the password.And after registration,login or recovery user will be redirected to the home page.

Home Page:
At the home page user will be able to see all task profiles that he has participated in. The user can sort and filter the task based on different categories and can focus on only the tasks he is needed to operate.

Task Details Page:
By clicking the details button in the task profile of home page the user will be redirected to the task details page.In the task details page the user can see the task todo status in the todo tab.He can perform comments in the comment section and also can see others comment. He can take notes in the note section. He can also see the task modification history to trace back a task status to a previous state.

My teams Page:
By selecting the my teams option from the left sidebar or from the top navigation menu in the small screen the user will be redirected to the My team page . Where the user will be able to see all the teams that they have participated in.

Team Details Page:
In the my teams page if a user clicks on a team then he will be redirected to the team details page where he will be able to see the details of the selected team . In the details page users can see the team chats,tasks,members,and team history. If a user is the team owner then he will be able to see the edit icon at the top right corner and upon clicking on it he will be able to fill up the team updation form and can save it.

Session Page:
By clicking on the create session button the user will be redirected to the session page where upon filling session details the user can start a session. But he can also generate the session details by the system which will consider the user's current tasks to generate the session. In the started session pages the use can take session notes as a form of outcome,can change the task list that has been operated at the session ,configure the break duration and intervals and so on.After completing the session the user can enter the end session button which will show a form to change or modify the session details and then finally he can save the session or justly simply not save the session.In the previous session page the user can see the details of the previously completed sessions. The user can navigate to this page by the previous page button.

Progress Report Page:
In left side bar by clicking the My progress button the user will be redirected to the progress report page where the user will see the progress of current tasks and its statuses based on different categories and visuals like bar chart , pie chart and so on.

Performance Report Page:
Similar to the progress report page but here the user can set the previous number or days considerations to generate the report.

Deep Work Analytics:
Similar to performance reports the user can see the deep working analytics based on the session data .

Account Dashboard Page:
By clicking on the profile icon at the nave bar the user can go to the profile dashboard where he can see and modify his account details as well as tasks and team details.

Notification,message,Feedbacks page:
In these pages the user can filter and search the respective contents.User can navigate to these pages by clicking on the icon on the navigation bar or left side bar for feedback or for small screens on the navigation section.

Challenges Faced:

Generating The Roadmap:
Creating the roadmap from the goal of the user was a challenging task. Like we had to maintain a template as a prompt which then is filled with the user input then this prompt passes to an ai model,getting the result in the actual json format Parsing the response and and send only the valid and expected json format was a difficult aspect of the project.

Session States Saving and reloading:
If a user closes a tab of a current session the user will be able to come back to that session's current state and don't have to create the session again. This is done by saving a copy of the session state and regaining it upon redirecting to the session route for a user. Keeping these things checked and performed was a tedious task to perform.

Team and Task Collaboration:
Sending the request to invite the user to the task or team and handling its response properly was a complex task to do given that invited member list and involvement of multiple notification handling.
Task access management:
All members of a team have access to the all team task. So giving access to the task of a newly joined team member or removing access to a left team member or removed team task was a sensitive and error prone part of our project which we performed with extra consideration as unauthorized user can get access to the task if not handled properly.

Future Scope And Conclusion:

In the given time span of the project until that part was done. But there can be room for improvement or adding some new features to the existing system to different modules. LIke in the session part, users can perform sessions collaboratively at a time instead of one. In the team apart from owner admin can also be added with special roles. Instead of tasks adding to the team a full roadmap could be added at a time.
In Conclusion the project is a useful task management tool for anyone who wants to handle daily activities in the form of some tasks and can increase their productivity by taking help of different other modules like team,session,analytics,communication to efficiently perform their task and become productive.

Reference:

Project Accessing URL:
https://spl-2-hmazf5wia-badhon1401s-projects.vercel.app
Project Source Code URL: https://github.com/Badhon1401/SPL-2
Next.Js Docs URL: https://nextjs.org/docs
Spring Boot Docs URL: https://docs.spring.io/spring-boot/index.html
Atlas URL:
https://www.mongodb.com/products/platform/atlas-database
Vercel site URL: https://vercel.com/
RailWay site URL: https://railway.com/
Mistral AI site URL: https://mistral.ai/
Shad CN UI Library Site URL: https://ui.shadcn.com/