WORKSPIRE

Software Requirements Specification

04.04.2025

Zeynep KURT 150121042

Damla KUNDAK 150121001

Ayşe Nisa ŞEN 150121040

Prepared for

CSE3044 Software Engineering Term Project

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to define the functional and non-functional requirements for the Team Collaboration and Productivity Web Application. The intended audience includes software engineers, developers, testers, project managers, and stakeholders involved in the project.

1.2 Scope

This software aims to enhance team collaboration, task management, and productivity within companies. Key features include:

- Feature assignment and tracking
- Individual and team dashboards
- Performance evaluation with scoring and ranking systems
- Built-in communication for messaging
- Pomodoro timer integration for time management
- Mini-game for engagement and motivation

This system will be a web-based application accessible via browsers, ensuring openness, responsibility, and efficient work processes within teams.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS**: Software Requirements Specification
- **UI**: User Interface
- **DBMS**: Database Management System
- **API**: Application Programming Interface
- **Pomodoro Timer**: A time management method based on 25-minute work intervals
- **Dashboard** : The main UI screen presented after login, containing key features and shortcuts
- **AI Task Prioritizer:** System feature that uses AI to suggest or sort tasks based on urgency or user behavior

- **Ranking / Scoreboard:** A system that tracks users' performance based on completed assigned tasks
- **Chat:** A communication feature that allows team members to message each other in real-time.
- **Smart Meeting Scheduler:** AI-powered feature that finds optimal time slots for meetings based on calendar availability.
- **Authentication:** The process of verifying a user's identity before granting system access.
- **Employee:** A regular system user who can create personal tasks, communicate, and track progress.
- **Manager:** A user role with permissions to assign tasks to employees and view team activity.
- **System:** The Workspire platform, including backend services and AI engines.

1.4 References

- *IEEE Standard for Software Requirements Specifications (IEEE 830-1998)*
- *Lecture Slides – CSE3044 Software Engineering*
- *Agile Software Development Principles*
- *PostgreSQL, Node.js, and React.js Documentation.*
- *React & Node.js Official Docs*

1.5 Overview

This document outlines the software's general description, specific requirements, and UML diagrams to provide a clear understanding of the system's functionalities.

2. General Description

2.1 Product Perspective

This application is a standalone web-based system designed for enterprise use. It integrates with existing project management tools through APIs and can be accessed via desktop.

2.2 Product Functions

- User authentication and role-based access
- Dashboard for managing tasks and progress
- Task assignment with deadlines
- Performance tracking with rankings
- Messaging and team communication
- Pomodoro timer for focus sessions
- Mini-game for employee engagement
- Smart meeting suggestions
- Task prioritization with AI
- Calendar and meeting integration
- Conflict detection and resolution

2.3 User Characteristics

- **Employees**: Utilize task management and communication features
- **Team Leads(Manager)**: Assign tasks and track team performance

2.4 General Constraints

- Web-based only
- Requires an internet connection

2.5 Assumptions and Dependencies

- Users will have basic familiarity with web-based project management tools.
- System integrates with PostgreSQL, Node.js, and React.js.
- Data security and encryption measures will be implemented.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

- Interactive dashboards
- Task tracking UI
- Chat interface for messaging

3.1.2 Hardware Interfaces

- Runs on standard web servers
- Requires devices with an internet browser

3.1.3 Software Interfaces

- Database: PostgreSQL
- Backend: Node.js
- Frontend: React.js

3.1.4 Communications Interfaces

- WebSocket for real-time messaging
- RESTful API for data transactions

3.2 Functional Requirements

3.2.1 User Login

- The system shall provide a login interface as the main access point for users.
- When users visit the platform, they shall see a form containing fields for email and password.
- Users must enter valid credentials. The email should match an existing user account stored in the database.
- After clicking the login button, the system will check the credentials. If they are valid, the system will generate a session token using JWT (JSON Web Token).
- If the credentials are incorrect, the user will receive an error message such as “Incorrect email or password.”
- Passwords will be encrypted using bcrypt and stored securely. No plaintext password storage is allowed.
- After a successful login, users will be redirected to the dashboard page.
- The login session will be stored using the browser’s localStorage to maintain user state.
- If a logged-in user navigates to the login page again, they will be automatically redirected to the dashboard.
- In case of any server or network issue, the system will show a notification and suggest the user to try again.

3.2.2 Add Personal Task

- The system shall allow logged-in users to create and manage their own personal tasks independently from team or assigned tasks.
- A button or visible input component shall be present in the user’s dashboard to initiate task creation.
- The task creation form shall include at least the following fields: Title (required), Description (optional), and Deadline (optional).

- Upon submitting the form, the system shall validate the input. The title field must not be empty; if it is, the user shall receive a warning message.
- After successful validation, the task shall be saved to the database, associated with the user's unique ID.
- The newly added task shall appear immediately in the personal task list through a UI update using React state.
- Users shall be able to mark tasks as completed, edit task details, or delete them entirely.
- Each task shall have a visual status indicator (e.g., "To Do," "In Progress," "Completed").
- If the server fails to save a task due to a connection or backend error, the user shall receive an error message with an option to retry.

3.2.3 Performance Evaluation

- Scoring and ranking based on task completion.
- Performance dashboards for individuals and teams.

3.2.4 Messaging System

- Real-time chat between team members.
- File sharing and message history.

3.2.5 Pomodoro Timer

- The system shall provide a built-in Pomodoro Timer that users can access from their dashboard interface.
- The timer shall follow the standard Pomodoro technique: 25 minutes of focused work followed by a 5-minute break.
- Users shall be able to start, pause, reset, and skip break intervals via clearly labeled buttons in the UI.
- When the user starts a Pomodoro session, the timer countdown shall be displayed in real-time on the screen.

- The UI shall include visual feedback, such as a progress animation, to indicate the remaining time.
- Users shall have the option to link a Pomodoro session to a personal task, allowing time-tracking for that specific task.
- When a Pomodoro session is completed, the system shall notify the user with a sound or visual pop-up and offer the option to start the break.
- Completed Pomodoro sessions shall be recorded and stored in the database with timestamps and linked tasks (if any).
- If the user navigates away or refreshes the page during an active session, the timer state shall persist using localStorage or an equivalent mechanism.
- The system shall prevent users from running multiple Pomodoro sessions at once to maintain focus.
- In case of browser crash or logout, previously recorded Pomodoro sessions shall still be available from the database for reporting or statistics.
- The UI shall be responsive and support both dark and light themes inherited from the layout context.

3.2.6 Mini-Game

- Simple game for engagement and motivation.

3.2.7 Smart Meeting Suggestion

- The system shall allow users to request a smart meeting time suggestion by selecting participants.
- Upon request, the system shall analyze each participant's calendar events and task load to find common available slots.
- The suggestion engine shall avoid conflicts with existing meetings and high-priority tasks.
- The system shall return a ranked list of suitable time slots, including start/end time and duration.

- Users shall be able to accept a suggested time, request more options, or manually adjust the meeting time.
- Once accepted, the meeting shall be scheduled and added to the calendars of all selected participants.
- All participants shall receive notifications about the meeting details.
- If no common time slot is found, the system shall provide the closest available alternatives and allow overriding optional restrictions.

3.2.8 Conflict Detection and Resolution

- The system shall detect scheduling conflicts between meetings and tasks.
- In case of a detected conflict, the system shall notify the user and offer rescheduling options.
- The user may choose to override the conflict warning and proceed.
- Conflict checks shall run periodically and after any new task or meeting is added.

3.3 Non-Functional Requirements

3.3.1 Performance

- The system shall respond to user actions such as login, task creation, and chat messaging in a reasonable amount of time under normal conditions.
- The backend and database shall be optimized to prevent noticeable delays in task loading or meeting suggestions.
- Real-time features (e.g., Pomodoro, messaging) shall work without noticeable lag during standard usage.

3.3.2 Reliability

- The system shall save user data accurately and consistently, even during temporary disruptions like page reloads or brief network issues.
- Core actions (e.g., saving a task, marking it as completed) shall not result in data loss or duplication.
- In the case of errors, users shall be informed and given a chance to retry their actions.

3.3.3 Availability

- The system shall be accessible to users during normal working hours.
- In the event of maintenance or downtime, users shall be shown an appropriate message or notification.
- Data stored in the database shall remain accessible once the system is back online.

3.3.4 Security

- User credentials shall be securely stored using password hashing.
- All user input shall be validated to prevent injection attacks and common vulnerabilities.
- Authentication shall be enforced using secure methods like JWT.
- Access to specific features shall be based on user roles (e.g. Manager, Employee).

3.3.5 Maintainability

- The system shall use a modular code structure to allow future updates and improvements with minimal disruption.
- Clear naming conventions and code documentation shall be followed throughout the project.
- The application shall support environment-based configurations (e.g., development, production).

3.3.6 Portability

- The application shall run on major browsers and different screen sizes (desktop, tablet, mobile).
- Core functionality shall work across environments (local development, test, and production).

3.4 Inverse Requirements

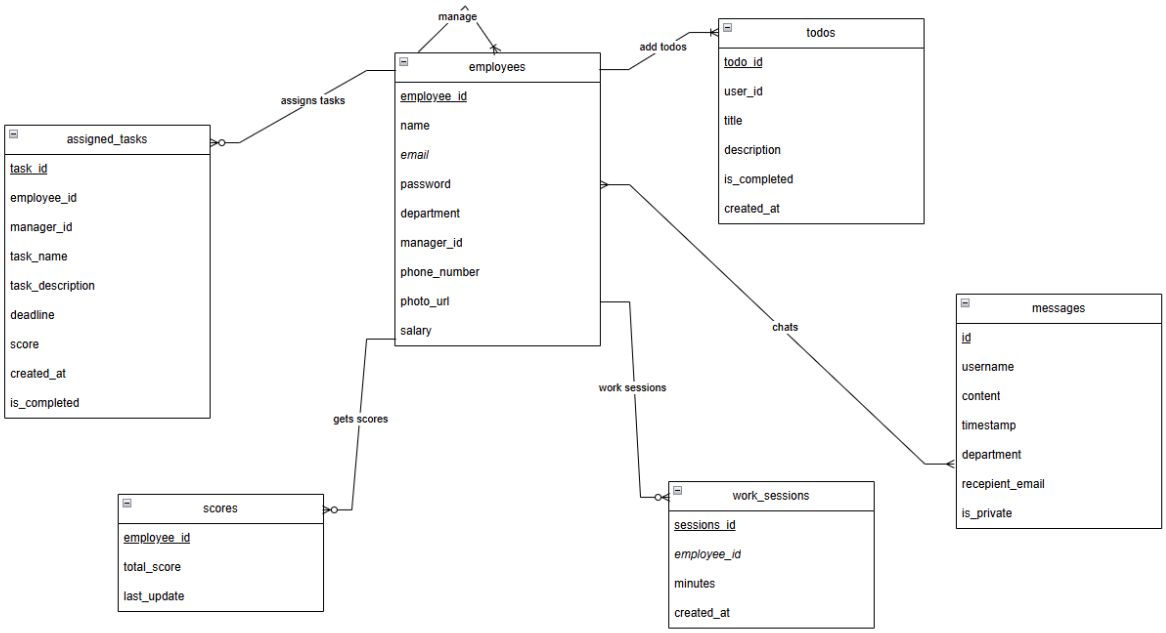
- The system shall not allow unauthorized users to access sensitive data.
- The mini-game shall not interfere with core functionalities.

3.5 Design Constraints

- Must use PostgreSQL, Node.js, and React.js.
- Comply with GDPR and data protection laws.

3.6 Logical Database Requirements

- Tables for users, tasks, messages, rankings, and timer logs.
- Relationships:



4. UML Diagrams

4.1 Use Cases

4.1.1 Use Case #1

User Login

Actor: Employee / Manager

Scope: WorkSpire System

Main Flow:

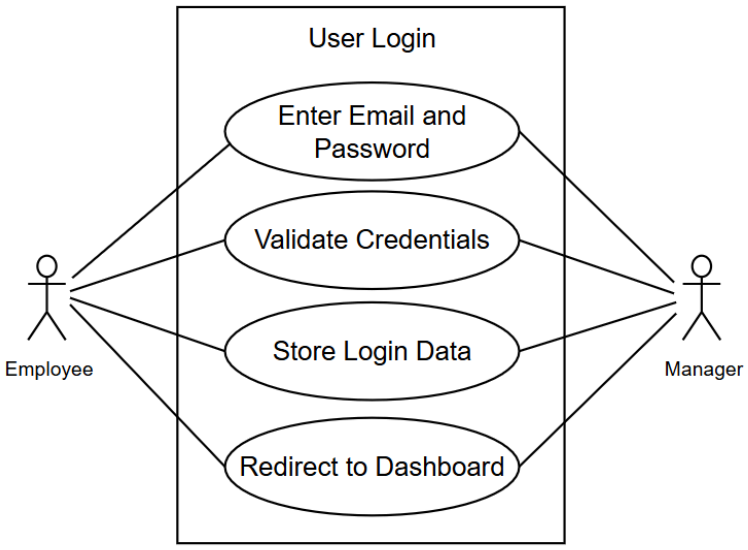
1. The user accesses the login page.
2. The login form is displayed with two input fields: Email and Password, and a Login button.
3. The user enters their email and password.
4. Upon clicking the Login button, the system:
    - Sends a POST request to the login endpoint (/login).
    - Verifies the credentials against the database.
5. If successful:

- The user receives a token.
- Relevant information is stored in localStorage (e.g., token, employeeId, managerId, department).
- The user is redirected:
  - to /manager if manager_id === 1
  - otherwise, to /home.

6. The dashboard loads according to the user's role.

Alternative Flows:

- Invalid Credentials:
  If the email or password is incorrect, the system displays an alert: “Incorrect email or password.”
- Missing Input:
  If either field is left empty, the system prevents submission and displays an appropriate warning.
- Backend or Network Failure:
  If the authentication service is unreachable, a system error message is displayed, prompting the user to try again later.



4.1.2 Use Case #2

Add Personal Task

Actor: Employee

Scope: WorkSpire Task Management System

Main Flow:

- Login:

  The Employee logs into the system using their credentials.

  Upon successful authentication, the system displays the Employee Dashboard.
- View Personal Tasks Section:

  The Employee sees a section titled “Personal Tasks” on the left panel of the dashboard.

  The section displays:
  - A text field labeled “Add new task”
  - A plus (“+”) button to submit a new task
  - A list of existing personal tasks with checkboxes and delete icons
- Add New Task:
  1. The Employee types a task name into the input field.
  2. The Employee clicks the + button to add the task.
  3. The system sends a POST request to the backend API /api/todos with the task data.
  4. The backend creates a new record in the todos table with:
     - task_name
     - is_completed set to false by default
     - associated employee_id
  5. The new task is rendered instantly in the task list under the personal task section.
- Optional Actions:
  - The Employee may check the box to mark a task as completed (handled in a separate use case).

- The Employee may click the red trash icon to delete a task. The system then sends a DELETE request to /api/todos/:taskId.

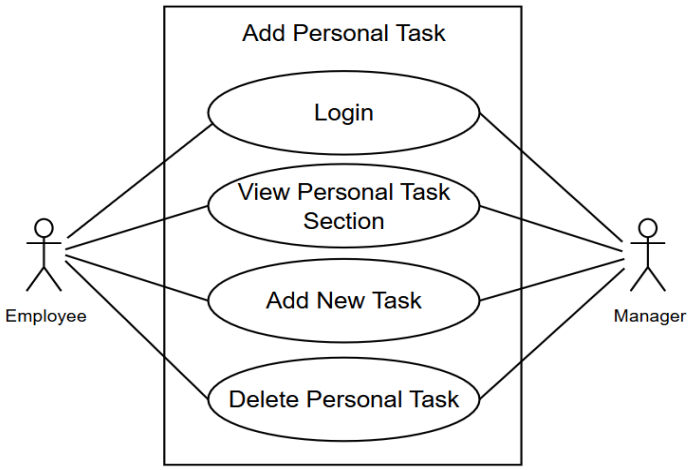
Alternative Flows:

- Empty Input Field:

  If the Employee clicks the + button without typing a task name:
  - The system prevents submission and shows a warning or ignores the click.
- Backend or Database Failure:

  If the task cannot be saved due to a system error:
  - An error message is displayed to the user.
  - The task is not added to the list.
- Unauthorized Action:

  If the Employee tries to add a task while not authenticated:
  - The system redirects to the login page or shows an authorization error.



4.1.3 Use Case #3

Assign Team Task

Actor: Manager

Scope: Workspire Task Management System

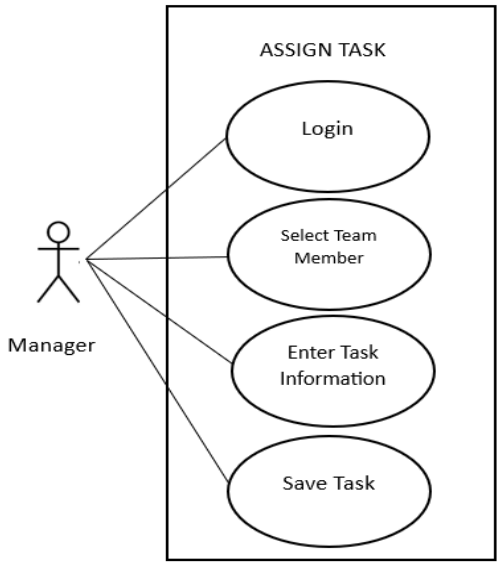
Main Flow:

1. Login:
    - The Manager logs into the system using their credentials.
    - If authentication is successful, the system displays the “Manager Dashboard”.
2. Select Team Member:
    - The Manager navigates to the "Assign Task" section on the dashboard.
    - The system retrieves and displays a list of team members reporting to the manager.
    - The Manager selects a specific user from the list.
3. Enter Task Information:
    - The Manager enters the following details in the task form:
        - Task Name (Mandatory)
        - Task Description (Optional)
        - Deadline (Mandatory)
        - Score (Optional - can be used for evaluation purposes)
    - The Manager submits the task form.
4. Save Task:
    - The system validates the task information.
    - The system saves the task to the assigned_tasks table in the database with the status as incomplete (by default).
    - The system associates the task with the selected user (team member) by storing their employee_id.
    - A success message is displayed confirming the task has been assigned.

Alternative Flows:

- Invalid Task Information:
    - If the task name or deadline is missing, the system displays an error message prompting the manager to enter the missing details.

- Network/Database Error:
    - If the system fails to save the task due to connectivity or database issues, an error message is displayed.
    - The system prompts the manager to retry or save the task later.



4.1.4 Use Case #4

Mark Task as Completed

Actor: Employee

Scope: Workspire Task Management System

Main Flow:

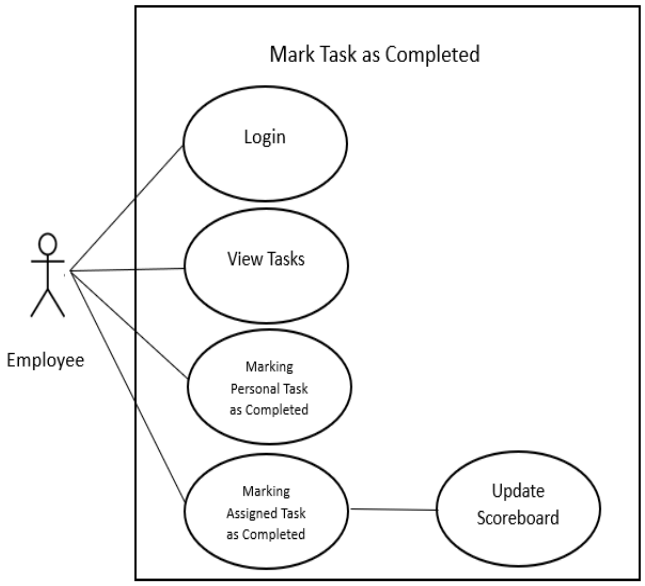
1. Login:
    - The Employee logs into the system using their credentials.
    - Upon successful authentication, the system displays the Employee Dashboard.
2. View Tasks:
    - The Employee sees two sections:
        - Personal Tasks (left panel)
        - Assigned Tasks (right panel)

    - Both sections display the tasks related to the employee.
3. Marking Personal Task as Completed:
    - The Employee checks the checkbox next to a personal task.
    - The system sends a request to the backend API (/api/todos/:taskId) to update the is_completed status to true.
    - The task is marked as completed in the database.
    - The task title is displayed with a strikethrough style on the dashboard.
4. Marking Assigned Task as Completed:
    - The Employee checks the checkbox next to an assigned task.
    - The system sends a request to the backend API (/employees/assigned-tasks/:taskId) to update the is_completed status to true.
    - The task is marked as completed in the assigned_tasks table.
    - The task title is displayed with a strikethrough style on the dashboard.
    - The task's score is added to the employee's total score in the Scoreboard table.
5. Update Scoreboard:
    - The backend processes the completed task and updates the scoreboard table by:
        - Fetching the employee's existing score.
        - Adding the score of the completed task to the total.
        - Saving the updated score to the database.
    - The updated score is reflected in the Scoreboard view of the application.

<u>Alternative Flows:</u>

- Failed Update:
    - If the database connection fails or there's a server error, the task completion status is not updated.
    - The user is prompted with an error message and encouraged to retry.
- Unauthorized Action:
    - If the employee attempts to mark a task they do not have permission for, an error message is displayed.

- No Tasks Available:
    - If the user has no tasks assigned, the respective section displays a message: “No tasks available”.



4.1.5 Use Case #5

Use Pomodoro Timer

Actor: Employee

Scope: WorkSpire Task Management System

Main Flow:

1) Login:

   The employee logs into the system using their credentials.

   Upon successful authentication, the system redirects the user to the dashboard.
2) Access Pomodoro Timer:

   The employee clicks on the "Pomodoro Timer" section.

   The screen displays the following elements:
   - A numeric input to define session duration (default: 25 minutes)
   - A “Set Time” button

- A countdown timer display
- Start and reset buttons
- A table titled “Weekly Work Time” showing session history

3) Start Timer:
    a) The employee enters a duration (e.g., 25 minutes).
    b) The employee clicks the “Set Time” button.
    c) The timer is set to the selected duration.
    d) The employee clicks the green “Start” button to begin the countdown.
    e) When the timer reaches 00:00, the session ends with a notification.
4) Save Session:

    The system logs the session data to the backend.

    The new session entry appears in the "Weekly Work Time" table.

    The data is also available for performance tracking and scoring.

Alternative Flows:

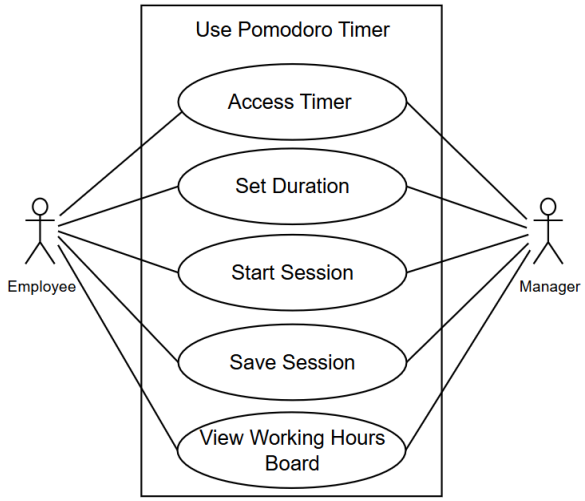
- Missing or invalid input:

  If the employee leaves the input field empty or enters an invalid value, the system either uses the default (25) or prompts the user to correct it.
- Resetting the timer:

  The employee clicks the blue reset button, which clears the countdown and allows a new session to be configured.
- Page refresh or navigation away:

  If the page is refreshed or closed during an active session, the timer resets unless local state is preserved.
- Data storage error:

  If the system fails to save the session data due to server or network issues, the user is notified and the session may not appear in the history.



4.1.6 Use Case #6

View Team and Company-wide Scoreboard

Actors

- Primary Actor: Employee and Team Lead
- Stakeholders and Interests:
    - o Employee: Seeks to view personal and team/company performance openly. Driven by performance comparison.
    - o Team Lead/Manager: Seeks to efficiently monitor team performance. Recognizes top and underperforming members.

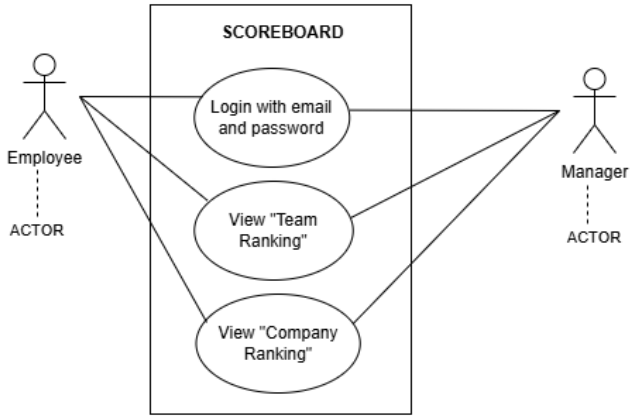
Main Success Scenario

1. The user enters their email and password on the login screen.
2. User is redirected to /home.
3. User opens the sidebar.
4. Clicks the “Scoreboard” option
5. By default, the Company view is active.

6. The employee list is sorted and rendered, displaying pictures, names, and scores.
7. User clicks the “Team” button to see the team’s score list.
8. The top three scorers are shown with larger pictures and trophy-style layout (centered gold for #1, etc.).

Alternative Flows

- API Error: If the backend API fails, a console error is logged, and no data is displayed.
- Empty List: If no scores are returned, only the empty table structure is rendered.
- Manager ID missing: If managerId is not in localStorage, fallback handling is required on the frontend.



4.1.7 Use Case #7

Real-Time Chat

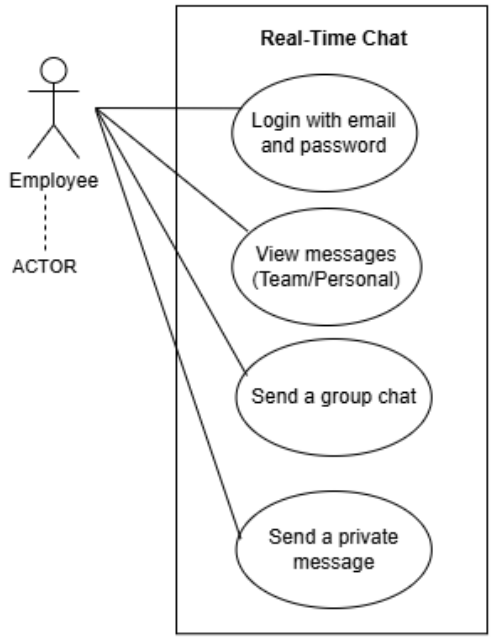
- Primary Actor: Employee
- Stakeholders and Interests:
    - Employee: Wants to communicate instantly with team members, either via group chat or private messages.
    - Manager: Monitors team communication, encourages collaboration.

Main Success Scenario:

1. Login:
    - User logs in through the login page
2. WebSocket Connection Established:
    - On page load, the frontend connects to the backend via Socket.IO.
    - Old messages are fetched and rendered in ChatContainer.
3. Group vs. Private Selection:
    - Sidebar (ChatSidebar) lists available private conversations and “Group Chat”.
    - User selects one to view message history in ChatWindow.
4. Sending a Message:
    - User types and sends a message.
    - Socket emits sendMessage with content, username, department, and recipient email.
5. Receiving Messages:
    - Other connected users receive the message via the same receiveMessage socket listener.
6. Chat Widget:
    - A floating chat widget is available across the web page for fast access.

Alternative Flows:

- If the recipient email is missing, the message is treated as a group message.
- If invalid email or user is not found, the server logs the event without crashing.



4.1.8 Use Case #8

AI Task Prioritizer

Actor:

- Primary Actor: Employee
- Secondary Actor: Team Lead (if they also use the AI prioritizer to view or assign tasks)

Scope:

Workspire Task Management System

Preconditions:

1. The user (Employee or Team Lead) is successfully logged in.
2. The system has access to relevant data sources:
    - User’s current list of tasks (with deadlines, priority, etc.).
    - Calendar events to detect possible conflicts or busy periods.

Main Flow:

1. Access Prioritizer Page
    a. The user navigates to a page or feature labeled “AI Task Prioritizer” from the dashboard or sidebar.
    b. The system fetches user’s tasks and calendar events from the backend to gather relevant data.
2. Task Analysis
    a. The system’s AI module analyzes each task’s deadline, complexity, and possibly current status (to-do, in-progress) as well as relevant calendar entries or meeting times.
    b. The AI may weigh tasks with imminent deadlines or higher assigned scores more heavily.

3. Priority
    a. The prioritizer produces a ranked list of tasks, from most urgent/critical to least, factoring in:
        i. Time until deadline
        ii. Overlapping events in the calendar
        iii. User’s existing workload or time availability
        iv. Task complexity or estimated duration
4. Display Results
    a. The system presents the user with a sorted list, highlighting tasks requiring immediate attention.
    b. Each task entry shows:
        i. Title, short description
        ii. Deadline
        iii. Recommended start date/time (if scheduling is possible)
        iv. Reason for the given priority (e.g., “Deadline within 1 day,” “High score impact,” etc.)

5. Accept or Adjust
    a. The user can accept the AI's recommendation, mark the top task as their next priority, or manually reorder tasks if they disagree with the AI's suggestion.

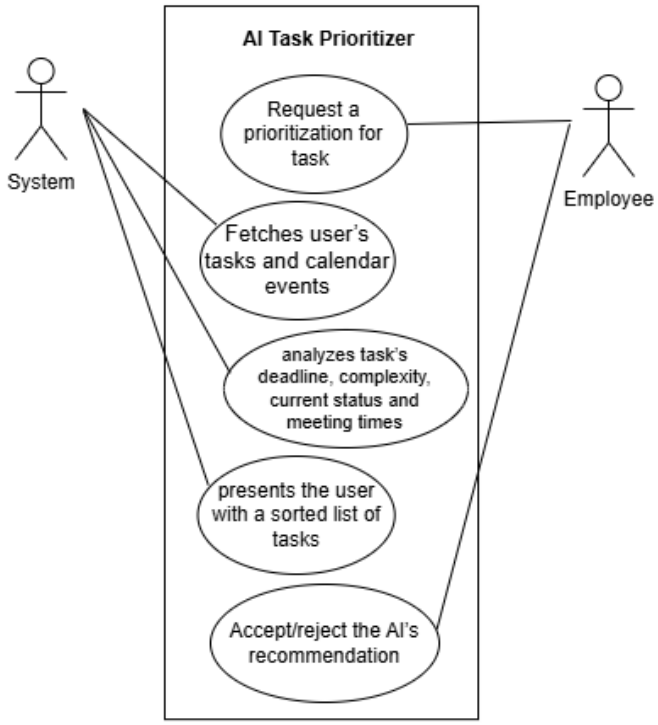
Alternative Flows:

1. No Tasks Available

   If the user has no tasks, the system shows a "No tasks found" message and prompts them to create one.
2. External Calendar Conflict

   If the AI detects that the user's time is fully booked or there is an unbreakable conflict, it warns the user.
3. Network/Server Error

   If at any point the AI module or database is not reachable, the system displays an error. The user can retry once the network is restored.



4.1.9 Use Case #9

Smart Meeting Suggestion

Actor: System (AI Suggestion Engine)
Scope: Workspire Task Management System

Main Flow:

1. User Requests a Meeting Suggestion:

- The user accesses the Smart Meeting Suggestion feature via the dashboard.
- The user selects the participants for the meeting.
- The user may optionally provide preferences (e.g., preferred time range, priority, duration).

2. System Fetches Data:

- The system retrieves the following data for each participant:
    - Personal Calendar Events: Existing meetings, tasks, or blocked time slots.
    - Task Deadlines: High-priority tasks that are due soon.

3. Data Analysis:

- The system analyzes:
    - Time slots where all selected participants are available.
    - Slots that do not conflict with high-priority tasks.
    - Time windows that best align with urgency and priority.

4. Generate Suggestions:

- The system generates a ranked list of optimal meeting time suggestions:
    - Start time, end time, and duration.
    - Compatibility with participants' priorities (e.g., avoids conflict with critical tasks).

5. Present Suggestions:

- The system displays the suggestions to the initiating user.
- The user can:
    - Accept a suggestion.
    - Modify the suggested time.
    - Request additional suggestions.

6. Schedule the Meeting:

- Once a suggestion is accepted:
    - The meeting is added to each participant's calendar.
    - Notifications are sent to all participants.
    - A confirmation message is shown to the user.

Alternative Flows:

• No Suitable Time Slot Found:

- The system notifies the user that no suitable time slot is available.
- The system provides the closest possible alternatives.
- The user may choose to override certain restrictions (e.g., schedule during non-critical task periods).

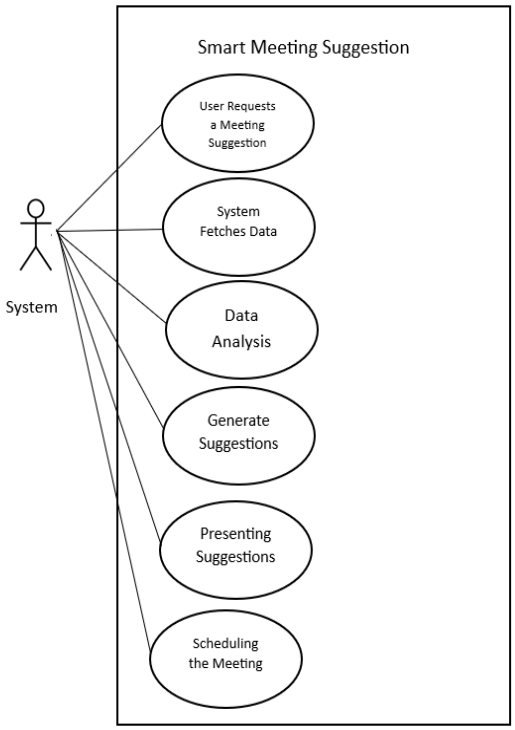
• Conflict with High-Priority Tasks:

- If a suggested time conflicts with a high-priority task, the system alerts the user.
- The user may choose to override or reschedule.

• Dynamic Rescheduling:

- The system periodically checks for conflicts between scheduled meetings and new tasks.

If conflicts arise, the user is notified and presented with updated meeting time suggestions.



4.2 Classes / Objects

4.2.1 Login

Attributes:

- email (string)
- password (string)
- error (string)

Functions:

- Capture user credentials
- Send login request to backend
- Redirect user upon successful login

4.2.2 TaskList

Attributes:

- tasks (array of personal tasks)
- userId (number)

Functions:

- Fetch personal to-do items from backend
- Display tasks in a list
- Show completion status

4.2.3 Scoreboard

Attributes:

- scores (array of employee scores)
- view (company or team)
- loggedIn, UserId

Functions:

- Fetch scoreboard data from backend
- Toggle between team and company views
- Highlight current user's score

4.2.4 MemoryGame

Attributes:

- cards, flippedCards, matchedCards
- turns, gameWon

Functions:

- Handle card flipping

- Check for matches
- Detect game completion

4.2.5 Pomodoro

Attributes:

- minutes, seconds
- isRunning, isBreak

Functions:

- Start, pause, and reset Pomodoro timer
- Track work and break intervals
- Optionally link timer to task

4.2.6 ChatBox / ChatPage

Attributes:

- messages, newMessage, recipient, department

Functions:

- Fetch and display messages
- Send new message via socket
- Handle department-based filtering

4.2.6 Calendar

Attributes:

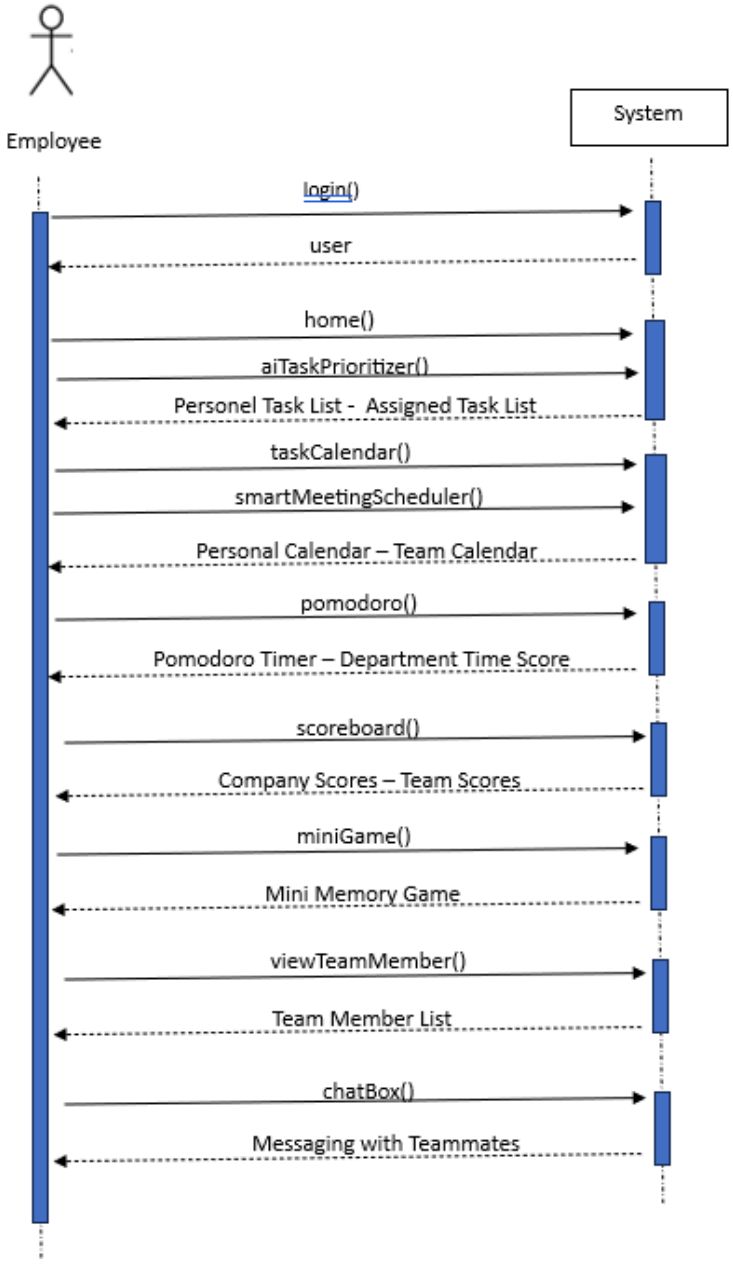
- viewMode (string)
- selectedMonth (number)
- selectedYear (number)
- currentWeek (dayjs object)
- meetings (array)

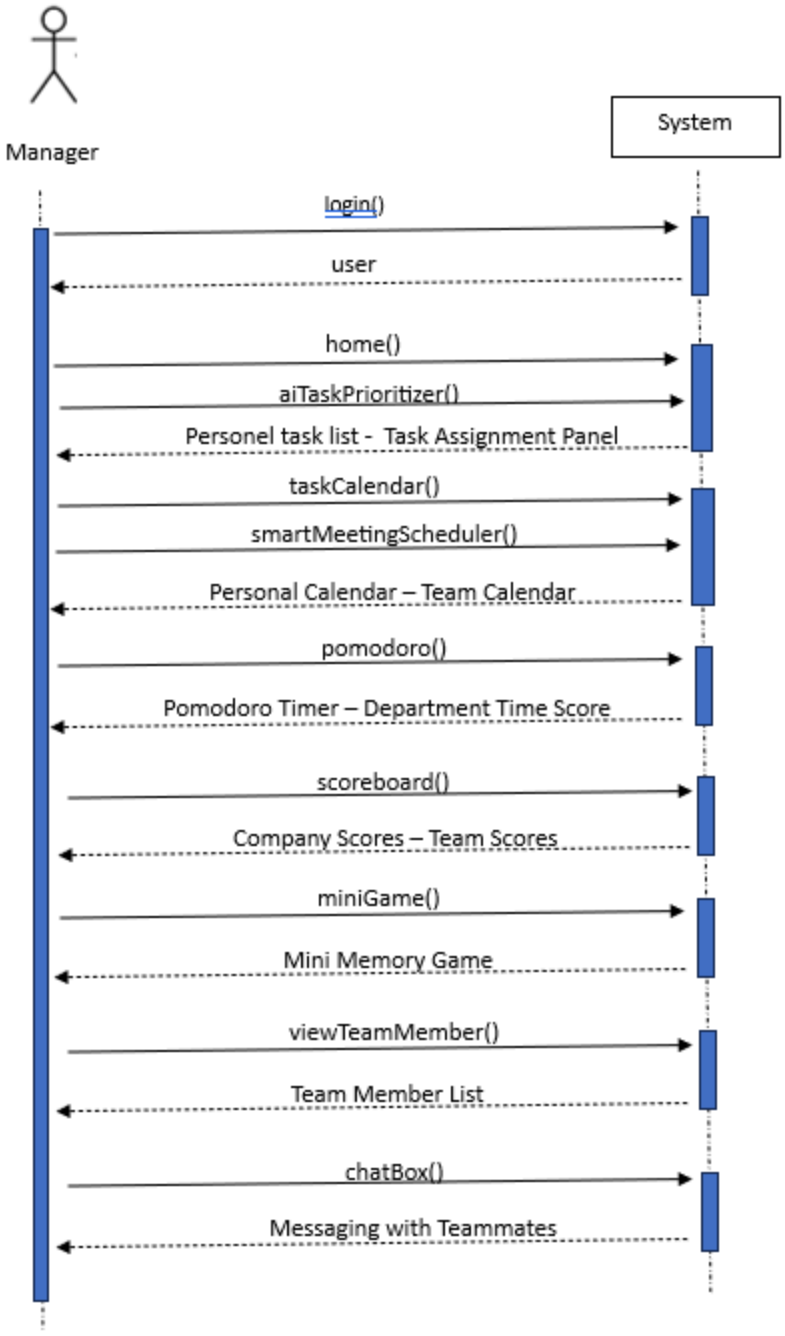
- tasks (array)
- newTaskTitle, newMeetingTitle (string)
- newMeetingParticipants (string)
- newTaskDate, newTaskTime (string)

Functions:

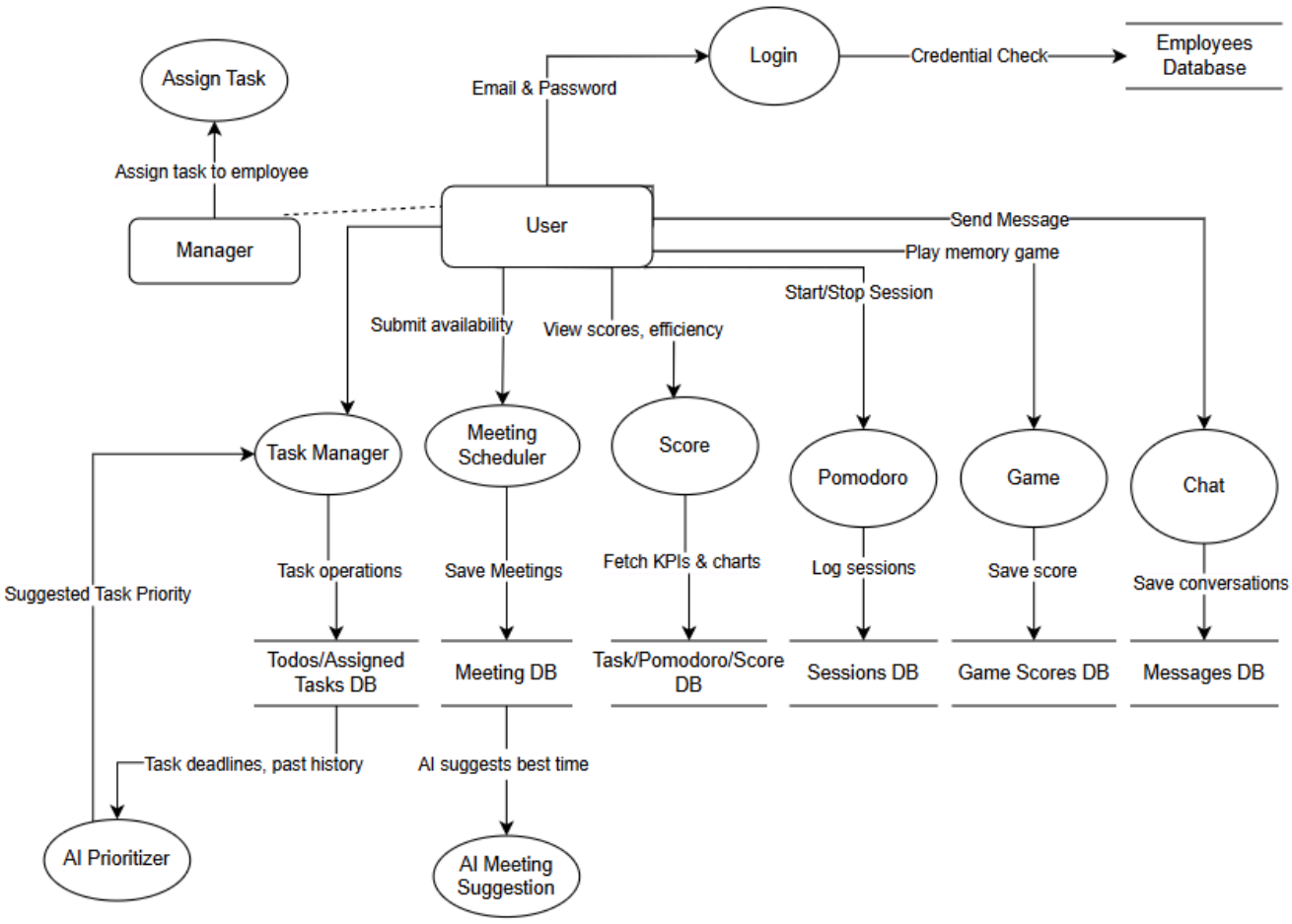
- Fetch tasks and meetings for the selected week
- Display tasks and meetings by day
- Add a new personal task
- Add a new team meeting
- Toggle between personal and team calendar views
- Navigate to previous or next week
- Filter tasks and meetings by selected day

4.3 Sequence Diagrams

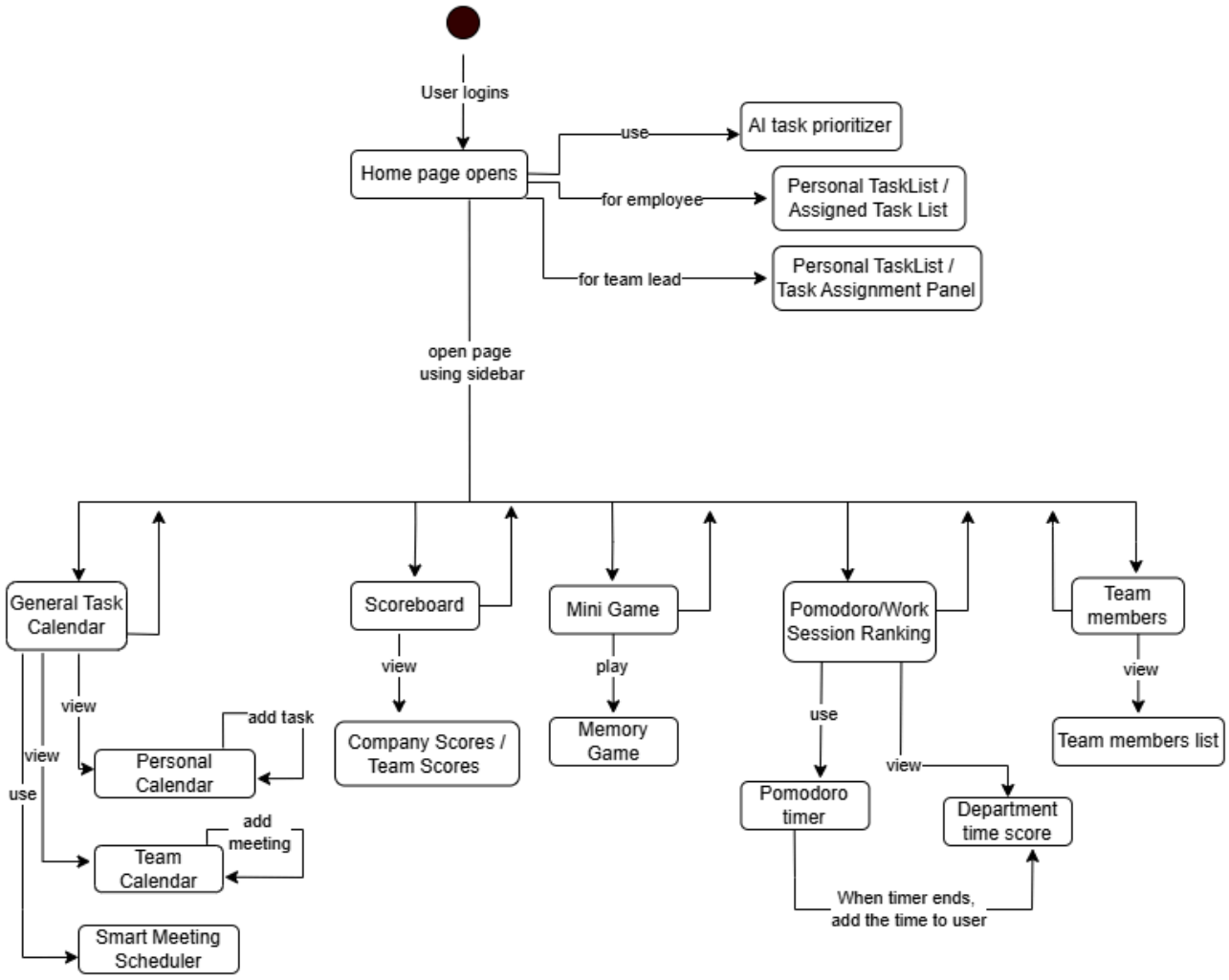



Manager
System
login()
user
home()
aiTaskPrioritizer()
Personel task list - Task Assignment Panel
taskCalendar()
smartMeetingScheduler()
Personal Calendar – Team Calendar
pomodoro()
Pomodoro Timer – Department Time Score
scoreboard()
Company Scores – Team Scores
miniGame()
Mini Memory Game
viewTeamMember()
Team Member List
chatBox()
Messaging with Teammates

4.4 Data Flow Diagrams (DFD)



4.5 State-Transition Diagrams (STD)



Work Distribution

1.1 Purpose 1.2 Scope 1.3Definitions,Acronyms,and Abbreviations 1.4 References 1.5 Overview 2.1 Product Perspective 2.2 Product Functions 2.3 User Characteristics 2.4 General Constraints 2.5 Assumptions and Dependencies 3.1 External Interface Requirements 3.2 Functional Requirements 3.3 Non-Functional Requirements 3.4 Inverse Requirements 3.5 Design Constraints 3.6 Logical Database Requirement 3.7 Other Requirements 4.1 Use Cases 4.2 Classes / Objects	Zeynep KURT Damla KUNDAK Ayşe Nisa ŞEN
4.3 Sequence Diagrams	Ayşe Nisa ŞEN
4.4 Data Flow Diagrams (DFD)	Zeynep KURT
4.5 State-Transition Diagrams (STD)	Damla KUNDAK