Converge

SOFTWARE ENGINEERING(IT-314)

Prof. Saurabh Tiwari

Group - 23

- 202301209- Gori Faran Firozbhai (Leader)
- 202301224- Prajapati Anushka Maheshbhai
- 202301244- Atharv Shah
- 202301213- Kalriya Prayag Jagdishbhai
- 202301251- Vasani Devarsh Chintanbhai
- 202301243- Vadaviya Chaitri Hareshkumar
- 202301212- Patel Ishti Upendra
- 202301211- Rathod Harita Rajeshbhai
- 202301228- Patel Naitikkumar Dilipbhai
- 202301242- Tejani Charvik Bipinbhai

Functional Requirements:

1. Authentication & Access Control

• FR-1: The system shall provide secure registration and login for all users.

• FR-2: The system shall support role-based permissions (Team Leader, Team Member) to control access to features and data.

2. Task & Workflow Management

• FR-3: The system shall allow “team leaders” to create and assign tasks with deadlines, priorities, and deliverables.

• FR-4: The system shall allow “team members” to view assigned task details, including title, description, priority, dependencies, and deadlines.

• FR-5: The system shall allow “team members” to update task status (e.g., “In Progress,” “Pending for approval”).

• FR-6: The system shall allow “Team leaders” to approve tasks and change status to completed.

• FR-7: The system shall support linking tasks with dependencies and visually indicate blocked tasks.

• FR-8: The system shall provide automatic task reminders for both “team members” and “Team leaders” before deadlines.

3. Project Tracking & Visualization

• FR-9: The system shall provide dashboards for team members and leaders to view project progress, task completion, and milestones.

• FR-10: The system shall allow “Team leaders” to generate progress reports in a clean and exportable format.

4. Communication & Collaboration

• FR-11: The system shall provide a real-time team chat system, both one-to-one and also group chat for “team members” and “team leaders”.

• FR-12: The system shall allow features such as searching chat history, organizing chats into threads, sharing files in chat.

• FR-13: The system shall generate daily chat summaries with key action items.

5. Calendar & Scheduling

• FR-14: The system shall provide a shared team calendar showing meetings, deadlines.

6. File Storage & Version Control

• FR-15: The system shall provide centralized file storage organized by project where users can upload, access and download files.

• FR-16: The system shall maintain version control for project files with ability to revert to previous versions.

7. Integration & Automation

• FR-17: The system shall integrate with third-party tools such as GitHub (Repository).

• FR-18: The system shall support AI chatbot assistance to provide guide on particular project.

Non-Functional Requirements:

• NFR-1: Performance: The platform must be responsive and support multiple concurrent users without significant lag.

• NFR-2: Scalability: The architecture must be able to handle a growing number of users and projects.

• NFR-3: Security: All user data and communications must be secured with end-to-end encryption.

• NFR-4: Privacy: User data will not be shared with any third parties without explicit consent.

• NFR-5: Availability: The system must remain consistently available with minimal downtime for core features like task management, communication, and file sharing.

• NFR-6: Usability: The user interface should be easy and simple to adapt.

• NFR-7: Cross-platform Sync: The system shall stay consistent across different devices (e.g. mobile, desktops, etc)

User Stories:

Format:

Front of Card (Story),

Back of Card (Success & Failure)

Title	Story	Success	Failure
US1 - View Task Details	As a team member, I want to view all the details for my assigned tasks so I can understand my responsibilities and plan my work.	• Task details (title, description, deadlines, priority, and dependencies) are clearly visible and updated.	• Information is missing or outdated, leading to confusion and errors.
US-2: Mark Task as Done	As a team member, I want to mark a task as pending review so my team leader can review and approve it.	• The task moves to a "Pending Review" state. • I'm notified when it's approved or when changes are requested.	• The task status doesn't update, or I don't receive feedback, delaying the process.
US-3: Receive Task Reminders	As a team member, I want to receive automatic notifications about upcoming deadlines so I don't miss them.	• I receive timely reminders for due tasks, allowing me to plan my time effectively.	• Reminders are not sent, or they're sent too late, causing missed deadlines.
US-4: View Task Deadline	As a team member, I would like to see deadlines for my tasks so that I can plan my work accordingly.	• Deadlines for all my tasks are clearly visible.	• Deadlines are missing, outdated, or inconsistent, causing me to miss important due dates.
US-5: Task Dependencies	As a team member, I would like to see task dependencies so that I know which tasks must be completed before starting mine.	• Task details clearly show predecessor/successor tasks.	• Dependencies are not displayed or are wrong; I start work on a task that's blocked or incomplete.
US-6: View Project Status	As a team member, I want to view the overall project status so I can understand how my tasks contribute.	• I can see the project's overall progress.	• Project status is unclear or outdated, leaving me unsure of my task's relevance.
US-7: Real-time Chatting	As a team member, I want to chat with my team in real time so we can quickly discuss tasks and resolve doubts without delays.	• I can instantly send and receive messages.	• Messages are delayed or lost, causing slow responses and confusion.
US-8: Chat Search	As a team member, I want to search through the chat history so I can easily find past conversations and information.	• I can find relevant messages by keywords and if not exist then state clear message that matching search not found.	
US-9: Organized Chats	As a team member, I want chat messages to be organized by thread so I can keep related discussions together.	• Conversations on specific message are threaded, keeping them separate from unrelated messages.	• All conversations are mixed together, making it hard to follow specific topics.
US-10: File Sharing in Chat	As a team member, I want to share files directly within the chat interface so I can collaborate on documents with my teammates.	• I can upload and share various file types. • Others can easily download them.	• File uploads fail, for example, due to size limits.
US-11: Daily Chat Summaries	As a team member, I want to receive a daily summary of key discussions so I can quickly catch up after being away.	• The system generates a concise summary of key discussions.	• Asummary is not generated, or it lacks relevant information.
US-12: Shared Team Calendar	As a team member, I want to view a shared team calendar so I can keep track of all project deadlines and meetings in one place.	• The calendar shows all team-wide meetings, deadlines.	• Calendar data is unavailable or outdated.
US-13: Centralized File Storage	As a team member, I want to store all project files in a central location, so I can easily access and collaborate on them.	• I can upload, access, and download project files from a centralized repository.	• I get a "Storage limit exceeded" error, preventing me from uploading files.
US-14: Version Control	As a team member, I want to have a version control system for project files, so I can track modifications and revert to previous versions if needed.	• The system automatically saves a new version when I save changes.	• I can view a history of all changes and restore a previous version.

US-15: Approve/Request Changes on a Task	As a team leader, I want to approve a task or request changes so the task can be officially finalized or improved.	• I can approve a task, which changes its status to "completed".	• The system doesn't update the task status or notify the team member, causing delays and confusion.
US-16: Set Task Dependencies	As a team leader, I want to set dependencies between tasks so the workflow is organized and tasks are completed in the right order.	• I can easily link tasks as predecessor and successor. • The system visually indicates tasks.	• Dependencies fail to save or update, leading to incorrect task order and project delays.
US-17: Receive Task Reminders	As a team leader, I want to receive automatic notifications before and after a task's deadline so I can take proactive action to prevent or resolve delays.	• The system automatically notifies me when a task is approaching its deadline (e.g., 24 hours away) and when it has passed its deadline without being completed. • Notifications include task details like the name, assignee, and deadline.	• No notifications are sent, or they are sent too late and lack key details, making them inef fective.
US-18: View Overall Project Status	As a team leader, I want to view the overall project status so I can monitor progress, identify bottlenecks, and take timely action to keep the project on track.	• I can see a real-time overview of all tasks, milestones, and team member contributions.	• Project status is missing or outdated, preventing me from identifying issues early.
US-19: Visual Progress Dashboard	As a team leader, I want to have a visual dashboard so that I can easily interpret overall progrss of project.	• The dashboard shows a clear overview of project status, including visual charts.	• Data sync fails, and the dashboard doesn't show the latest updates
US-20: Calendar Integration	As a user, I want to integrate my calendar with external services like Google Calendar or Outlook so I can manage all my events in a single place.	• Events created in the app sync with my external calendar and vice versa, with real-time updates.	• Syncing fails or is delayed, causing missing events or outdated information.
US-21: Security	As a business owner, I want all data in the collaboration tool to be secure so that the confidential information is protected from unauthorized users.	• All files, messages, and tasks must be encrypted.	• Data is transmitted or stored without encryption.
US-22: Usability & Adoption	As a business owner, I want the tool to be simple so that my employees can adopt it quickly.	• UI must be clean and consistent across the web.	• UI is inconsistent across different platforms.
US-23: Performance & Responsiveness	As a business owner, I want the collaboration tool to load and respond quickly so that work is not slowed down during projects.	• Pages should load within a few seconds under normal network conditions and support enough concurrent users without lag.	• The system crashes when multiple users are online.

Use Cases and Scenarios:

1. Use Case: Login / Register

Goal: The user wants to log in to the system if they already have an account, or register as a new user to start using the platform.

Actors: Team Member, Team Leader

Preconditions:

- The user has access to the platform.
- If logging in, the user already has a registered account.
- If registering, the user must have a valid email ID.

Postconditions:

- The user is successfully logged in and taken to their dashboard.
- The system grants access based on the user's role (Team Leader, Team Member, or Freelancer).

Main Flow:

1. The user navigates to the login/register page.
2. The user selects either Login or Register.

Login

3a. The user enters their email/username and password.
4a. The system validates the credentials.
5a. If the credentials are correct, the system logs the user in.

Register

3b. The user completes the registration form with their name, email, password, and role.
4b. The system checks the details for validity.
5b. If the details are valid, the system creates the new user account.

3. After successful login or registration, the user is redirected to their personalized dashboard.

Exception Flows:

- Incorrect Credentials: If login credentials are incorrect, the system displays an error message and prompts the user to try again.
- No Account on Login: If a user attempts to log in without an account, the system suggests they register.
- Already Registered Email: If a user tries to register with an email that's already in use, the system notifies them and suggests they log in instead.
- System/Network Issue: If the login or registration process fails due to a server/network issue, an error message appears, and the user is asked to try again later.

2. Use Case: Create Project

Goal: The user (Team Leader or Freelancer) wants to create a new project workspace to organize tasks, files, and communication for the team.

Actors: Team Leader

Preconditions:

- The user is logged into the system.
- The user has permission to create a project (Team Leaders always can).

Postconditions:

- A new project workspace is created in the system.
- The creator (Team Leader) becomes the admin of that workspace.
- An invitation request is sent to add other team members.

Main Flow:

1. The user logs into the system and navigates to the dashboard.
2. The user clicks on “Create New Project.”
3. The user enters the project details, including the project name, description, start date, and deadline.
4. The user sets roles and permissions for potential team members.
5. The user confirms the creation of the project.
6. The system creates the project workspace and generates a unique invitation link.
7. The system displays the new project dashboard, allowing the creator to start adding tasks and inviting members.

Exception Flows:

- Missing Information: If the user doesn't fill in required fields (like the project name), the system shows an error and prompts them to complete the form.
- Duplicate Project Name: If the project name already exists, the system displays an error and suggests a different name.
- System/Network Error: If the system fails to create the project due to a server or network issue, an error message appears, and the user is asked to try again later.

3. Use Case: Add Team Members

Goal: The Team Leader wants to add new members to the project workspace..

Actors: Team Leader

Preconditions:

- The Team Leader is logged into the system.
- A project workspace has already been created.
- For adding: The Team Leader has an invitation link ready to send.

Postconditions:

- If adding, the invited Team Member successfully joins the workspace and can access tasks, files, and chat.

Main Flow:

1. The Team Leader navigates to the project workspace.
2. The Team Leader goes to the “Members” section and selects “Add Member.”
3. The system generates a unique invitation link.
4. The Team Leader sends the link to the new Team Member (via email or messaging).
5. The Team Member accepts the invitation link, logs in, and is automatically added to the workspace.

Exception Flows:

- Invalid Invitation Link: If the invitation link is expired or invalid, the system displays an error and prompts the Team Leader to generate a new link.
- Member Already Exists: If the invited person is already a member of the workspace, the system notifies the Team Leader that no action is needed.
- Unauthorized Removal: If the Team Leader tries to remove themselves or another team leader, the system shows an error and prevents the action.
- System/Network Error: If a system or network failure occurs during the add process, an error message appears, and the user is asked to try again later.

4. Use Case: Personalized Chatbot

Goal: The user interacts with an AI-powered chatbot to get quick suggestions, or answers to queries related to the project or system.

Actors: Individual User (Team Member, Team Leader)

Preconditions:

- The user is logged into the system.
- The chatbot service is active and available.

Postconditions:

- The user receives relevant responses from the chatbot.
- The chatbot may provide suggestions.

Main Flow:

1. The user opens the chatbot from the workspace dashboard.
2. The user types a question or request.
3. The chatbot processes the input using its AI engine.
4. The chatbot fetches relevant information from the system, such as tasks or project data.
5. The chatbot replies with a response, like a task list, a reminder confirmation, or a helpful suggestion.

Exception Flows:

- Unrecognized Query: If the chatbot doesn't understand the question, it asks the user to rephrase or suggests related options.
- No Data Found: If the user asks for information that doesn't exist, the chatbot informs them.
- System Error: If the chatbot cannot fetch data due to a system or server issue, it displays an error and asks the user to try again later.
- Limited Permissions: If the user requests an action outside of their role's permissions (e.g., a team member asking to remove a member), the chatbot explains that the action is not allowed.

5. Use Case: Join Project Workspace

Goal:
Team member wants to join the project workspace to start collaboration.

Actors:
Team Member

Preconditions:

- Team member is logged into the system.
- Team leader has already created a project and sent an invitation link.

Postconditions:

- Team member has access to the project workspace, including tasks, files, and chat.
- Team leader can see that the team member successfully joined the workspace.

Main Flow:

1. Team Leader creates a new project workspace and sends an invitation link to the Team Member.
2. Team Member receives the link via email or internal message.
3. Team Member clicks the invitation link.
4. Team Member logs into the platform (if not already logged in).
5. System authenticates the user and validates the invitation link.
6. Team Member is added to the project workspace.
7. The workspace dashboard opens, showing tasks, files, and chat.

Exception Flows:

- Invalid/Expired Invitation Link: If the invitation link is invalid or expired, the system shows an error message and suggests requesting a new link.
- User Not Authenticated: If the team member is not logged in, the system redirects them to the login page before proceeding.
- Already Joined: If the team member has already joined the workspace, the system notifies them and directly opens the workspace.
- System/Network Error: If the system fails to add the member due to a server/network issue, an error message appears, and the user is asked to try again later.

6. Use Case: Assign Task to Team Member

Goal:
Team leader wants to assign a task to a specific team member.

Actors:
Team Leader

Preconditions:

- Team leader is logged into the system.
- A project workspace is already created.
- Team member has joined the project.

Postconditions:

- Task is assigned and appears in the team member's task dashboard.
- Notification is sent to the assigned team member.

Main Flow:

1. Team Leader logs into the system and opens the project workspace.
2. Navigates to the 'Tasks' tab.
3. Clicks **Create Task** and fills in task name, description, deadline, and assignee.
4. Selects a specific Team Member from the dropdown.
5. Clicks **Assign.**
6. Team Member receives a notification that a new task has been assigned.
7. Task appears in the Team Member’s dashboard.

Exception Flows:

- Team Member Not Found: If the selected member is not part of the project, the system shows an error message.
- Missing Details: If task details (name, deadline) are not filled, the system prompts the user to complete all fields.
- System Error: If the task cannot be created due to a backend error, the system displays an error message.

7. Use Case: Track Task Progress

Goal:
Team member wants to update task status and track their progress.

Actors:
Team Member

Preconditions:

- Team member is logged into the system.
- Task has been assigned to them.

Postconditions:

- Task status is updated (e.g., In Progress, Completed).
- Team leader can view the updated progress.

Main Flow:

1. Team Member logs into the platform.
2. Opens the assigned project workspace.
3. Navigates to the Tasks section.
4. Opens an active task to view details.
5. Clicks the status dropdown and selects In Progress.
6. After completing the task, updates the status to Completed.
7. Team Leader gets notified of the status change.

Exception Flows:

- Unauthorized Update: If the user tries to update a task not assigned to them, the system blocks the action.
- Invalid Status Change: If the transition is not allowed (e.g., skipping from *Not Started* to *Completed*), an error is shown.
- System Error: If the update fails, the system notifies the user to retry.

8. Use Case: Task Deadline Reminders

Goal:
Team member receives timely reminders about upcoming task deadlines to ensure tasks are completed on time.

Actors:
Team Member

Preconditions:

- Team member is logged into the system.
- Team member has assigned tasks with defined deadlines.

Postconditions:

- Team member receives a notification regarding an approaching task deadline.
- Team member is aware of the remaining time to complete the task.

Main Flow:

1. The System monitors all assigned tasks and their respective deadlines.
2. As a task deadline approaches (e.g., 24 hours before, 12 hours before), the system identifies the relevant team member.
3. The system generates a reminder notification.
4. Team member receives the notification through their preferred channel.

5. Team member views the notification, which includes details of the task and its deadline.

Exception Flows:

- Task Completed Early: If the task is marked as complete before the reminder is scheduled, the system cancels the reminder.
- System/Network Error: If there is a system or network issue preventing the delivery of the reminder, the system logs the error and attempts to resend the reminder when the issue is resolved.

9. Use Case: Version Control for the Project

Goal:
Users can manage and track different versions of project, ensuring changes are recorded, and previous versions can be restored.

Actors:
Team Member, Team Leader

Preconditions:

- Users are logged into the system.
- Project files are stored within the system's designated file management area.
- Version control is enabled for the project.

Postconditions:

- New versions of files are saved and accessible.
- Users can view the history of changes made to a file.
- Users can revert to a previous version of a file.

Main Flow:

1. Users navigate to the desired project workspace and access the file management section.
2. Users select a file for which they want to manage versions.
3. Users can either upload a new version of an existing file or view the version history.
4. If uploading a new version, the system prompts for a description of the changes.
5. The system saves the new version, incrementing the version number.
6. If viewing version history, the system displays a list of all previous versions, including the date, time, and user who made the changes, along with any change descriptions.
7. Users can select a previous version to view or restore.
8. If restoring, the system replaces the current version with the selected previous version, creating a new entry in the version history.

Exception Flows:

- **7a) No Previous Versions:** If a file has no previous versions, the system informs the user.
- **System/Network Error:** If there is a system or network issue preventing the saving, viewing, or restoring of file versions, an error message appears, and the user is asked to try again later.

- **Conflicting Changes (Manual Resolution):** If the user attempts to save a file with the existing name and type, then the system will ask the user to either cancel upload or overwrite.

10. Use Case: File Sharing - Add, Remove

Actors:
Team Member, Team Leader

Goal: Provide a reliable file sharing system where team members can upload, replace, and remove files associated with projects and tasks.

Preconditions:

- User is authenticated and has permission to perform file actions for the project or task.

Postconditions:

- Files are stored securely and linked to the task or project.
- Version history is available where supported.

Main Flow: (Add File):

1. Go to the project or task file section and click 'Upload' or 'Add File'.
2. Select the file, optionally add a description or tags, and confirm upload.
3. System validates file type and size and stores the file linked to the project or task.

Main Flow: (Remove File):

1. Select the file and click 'Delete' or 'Remove'.
2. Confirm deletion when prompted to prevent accidental loss.

Exception Flows:

- File Type/Size Rejected: Block uploads with unsupported types or sizes and inform the user.
- Permission Denied: Prevent file actions if the user lacks rights.

11. Use Case: Communication (Channels, Direct Messages)

Actors:
All Project Users - Team Members

Goal: Goal: Provide real-time communication across team group.

Preconditions:

- User is authenticated and part of at least one workspace.
- Real-time messaging service is available and configured.

Postconditions:

- Messages are delivered and stored according to retention rules.
- Mentions raise a notification to the mentioned user.

Main Flow:

1. Open the Chat/Communication tab and select a channel or direct message thread.
2. Type a message, attach files or snippets if needed.
3. Send the message.
4. System delivers the message in real time to online users.

Exception Flows:

- Message Delivery Failure: If real-time delivery fails, queue and retry and show delivery status to the sender.
- Spam or Abuse: Provide reporting and moderation for content that violates rules.
- Attachment Failure: If attaching a file fails, show an error and allow retry.