IT 314
Software Engineering

Real Time Collaborative Editor

Use Case Documentation

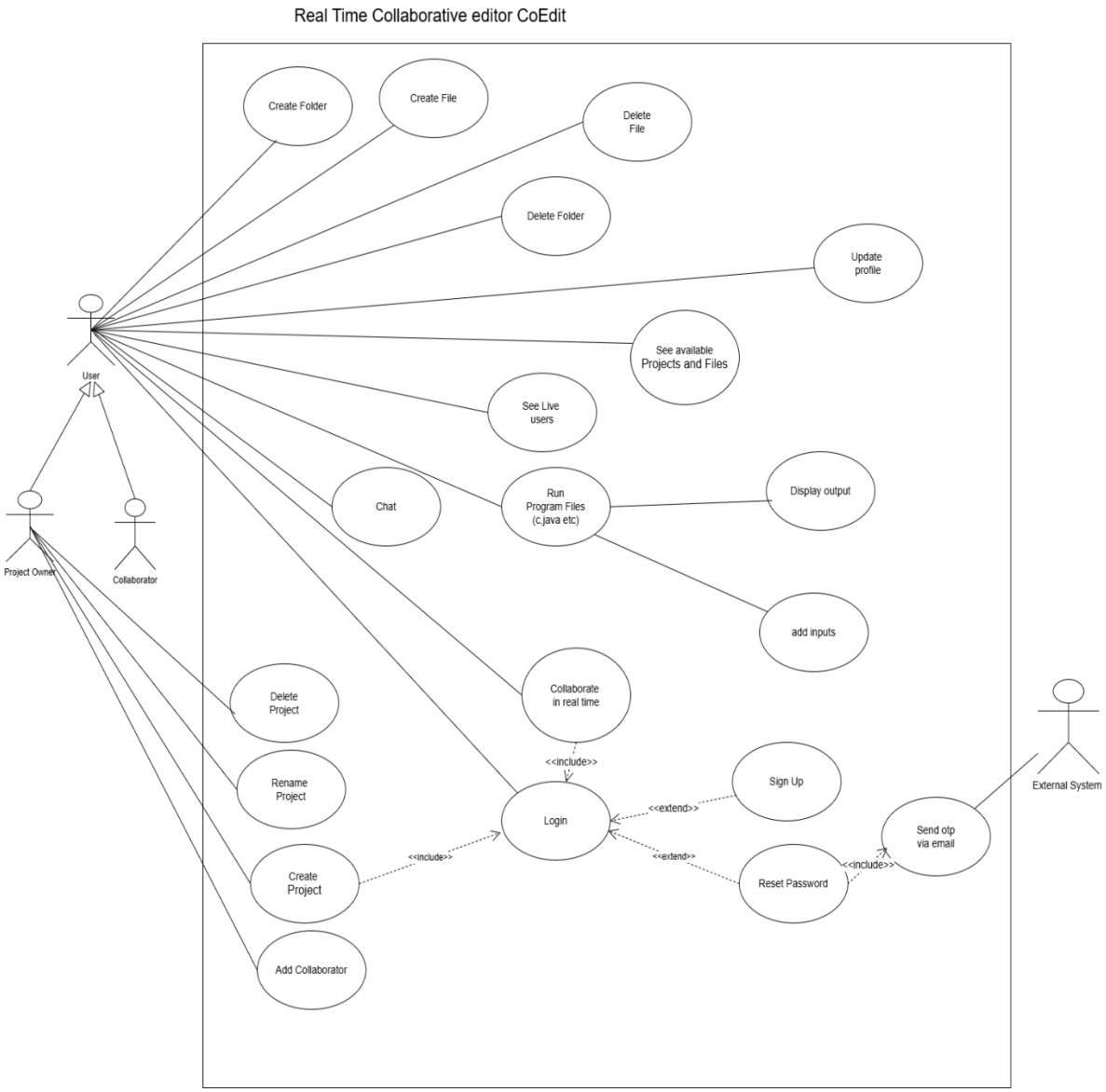
Dhirubhai Ambani Institute of Information and
Communication Technology
Gandhinagar, Gujarat

Submitted By
G35

Date of Submission
2/12/2024

Use Case Documentation :

Use Case Diagram :



Use Case Description:

Use Case 1: Login

Name : Login

Actor(s):

- User (Collaborator, Project Owner)

Preconditions:

- The user must have registered an account (manually or via Google).

Postconditions:

- The user is successfully logged in and gains access to the system's features.

Main Flow:

1. **User Action:** The user navigates to the login page.

   System Response: Displays the login form with the following options:
   - **Option 1:** Login manually using email and password.
   - **Option 2:** Login with Google.

2. **Option 1: Login Manually**
   - **User Action:** The user enters their registered email and password.
   - **System Response:** Verifies the credentials. If correct, redirects the user to the dashboard.

3. **Option 2: Login with Google**
   - **User Action:** The user clicks "Login with Google."
   - **System Response:** Redirects the user to the Google login page.
   - **User Action:** The user selects their Google account and grants permission.

- **System Response:** Authenticates the user and logs them into the system.

Alternative Flows:

- **Invalid Email/Password (Manual Login):**
  1. The system detects incorrect email or password.
  2. The system shows an error message.

- **Google Authentication Failed:**
  1. The user cancels Google login or the authentication fails.
  2. The system displays an error message.

Use Case 2: Sign Up

Name : Sign Up

Actor(s):
- External User

Preconditions:
- The user must not already have an account.

Postconditions:
- A new user account is created, and the user is able to log in.

Main Flow:

1. **User Action:** The user navigates to the "Sign Up" page.

   System Response: Displays the sign-up form with the following options:
   - **Option 1:** Sign up manually by entering username, email, and password.

- **Option 2:** Sign up using Google.

2. **Option 1: Sign Up Manually**
   - **User Action:** The user fills in the fields for username, email, and password, then submits the form.
   - **System Response:**
     - Validates the data (e.g. password is at least 8 character long).
     - Creates the account.
     - Redirects the user to the login page with a success message: "Account created successfully."

3. **Option 2: Sign Up with Google**
   - **User Action:** The user clicks "Sign Up with Google."
   - **System Response:** Redirects the user to the Google sign-up page.
   - **User Action:** The user selects their Google account and grants permission.
   - **System Response:**
     - Creates the account using the Google-provided email and username.
     - Redirects the user to the login page with a success message: "Account created successfully."

Alternative Flows:

- **Invalid Input (Manual Sign Up):**
  1. The system detects invalid input (e.g., weak password, non-unique email).
  2. The system shows an error message.

- **Google Authentication Failed (Google Sign Up):**
  1. The user cancels Google sign-up or the authentication fails.
  2. The system shows an error message.

Use Case 3: Reset Password

Name : Reset Password

Actor(s):

- User

Preconditions:

- The user must have a registered email address associated with their account.

Postconditions:

- The user's password is successfully reset, and they can log in using the new password.

Main Flow:

1. **User Action:** The user clicks the "Forgot Password" link on the login page.
   System Response: Displays a form to enter the registered email address.

2. **User Action:** The user enters their email and submits the form.
   System Response:
   - Verifies that the email exists in the system.
   - Sends an OTP (One-Time Password) to the provided email address.
   - Displays a form to enter the OTP.

3. **User Action:** The user retrieves the OTP from their email and enters it in the form.
   System Response: Verifies the OTP. If correct, displays a form to reset the password.

4. **User Action:** The user enters the new password and confirms it.
   System Response: Updates the password and displays a success message.

Alternative Flows:

- **Invalid Email:**
  1. The system detects that the entered email is not registered.
  2. The system shows an error message.

- **Invalid OTP:**
  1. The user enters an incorrect OTP.
  2. The system shows an error message.

Use Case 4: Create Project

Name : Create Project

Actor(s):

- Project Owner

Preconditions:

- The user must be logged in.

Postconditions:

- A new project is created and visible in the project list.

Main Flow:

1. **User Action:** The project owner selects the "Create Project" option.
   System Response: Displays a form to enter project details (name, description).

2. **User Action:** The user enters the details and submits the form.
   System Response: Creates the project and shows it in the user's project list.

Use Case 5: Add Collaborator

Name : Add Collaborator

Actor(s):

- Project Owner

Preconditions:

- The project must already exist.
- The collaborator must have a registered account.

Postconditions:

- The collaborator is successfully added to the project.

Main Flow:

1. **User Action:** The project owner selects the "Add Collaborator" option.
   System Response: Prompts the user to enter the collaborator's email.

2. **User Action:** The project owner enters the email and submits.
   System Response: Sends an invitation to the collaborator and grants them access to the project.

Use Case 6: Delete Project

Name : Delete Project

Actor(s):

- Project Owner

Preconditions:

- The user must have ownership rights for the project.

Postconditions:

- The project is permanently removed from the system.

Main Flow:

1. **User Action:** The project owner selects the "Delete Project" option.
   System Response: Displays a confirmation dialog.

2. **User Action:** The user confirms the deletion.
   System Response: Deletes the project and updates the project list.

Use Case 7: Real-Time Collaboration

Name: Real-Time Collaboration

Actor(s):

- Collaborator
- Project Owner

Preconditions:

- A project must already exist, and the user must have access to collaborate.
- The user must be logged into the system.

Postconditions:

- Changes made by one user are updated in the database and reflected for all collaborators in real time.

Main Flow:

1. **User Action:** A user opens a shared file or project for collaboration.
   System Response: Displays the file or project interface, along with the live user indicators showing who else is currently editing.

2. **User Action:** The user makes changes to the file (e.g., edits text, adds code, or modifies content).
   System Response:
   - Updates the changes in the database.
   - Propagates the changes to all other collaborators' interfaces in real time.

3. **User Action:** Other collaborators see the updated changes and continue editing collaboratively.
   System Response: Ensures all changes are synchronized across all active users.

Alternative Flows:

- **Network Issue:**
  1. A user's connection is interrupted during collaboration.
  2. The system temporarily queues their changes locally and syncs them with the database once the connection is restored.

- **Database Failure:**

1. If the database fails to update, the system notifies the user with an error message.

Scenario (User Action and System Response):

1. **User Action:** The first collaborator makes an edit to the document.
   System Response:
   - Immediately saves the changes to the database.
   - Notifies all other collaborators about the update and displays the changes in their view.
2. **User Action:** Another collaborator continues editing the document based on the updated content.
   System Response:
   - Follows the same process of updating the database and propagating changes.

Use Case 8: Chat

Name : Chat

Actor(s):

- Collaborator
- Project Owner

Preconditions:

- The user must be logged in and have access to the project.
- The user is within an active collaborative session.

Postconditions:

- The sent message is saved to the database and is displayed to all other collaborators in real-time.

Main Flow:

1. **User Action:** A user opens the chat interface in the project.
   System Response: Displays the existing messages and the input field to send new messages.

2. **User Action:** The user types a message and sends it.

System Response:

- Saves the message to the database.
- Immediately pushes the message to all active collaborators.
- The message appears in the chat window for all collaborators in real time.

3. **User Action:** Other collaborators see the message appear in their chat window.
   System Response: Displays the message to all other users who are currently in the chat.

Alternative Flows:

- **Network Issue:**
  1. A user's network connection is temporarily lost while sending a message.
  2. The system saves the message locally and re-sends it to the database once the connection is restored.
- **Database Failure:**
  1. If the message cannot be saved to the database due to an error.
  2.The system shows an error message:

Scenario (User Action and System Response):

1. **User Action:** The first collaborator types a message and sends it.
   System Response:
   - Saves the message to the database and displays it in the chat window.
   - Propagates the message to all other collaborators in real time.
2. **User Action:** Other collaborators see the message instantly in their chat interface and continue the conversation.
   System Response: The system continuously updates the chat interface for all active users with new messages.

Use Case 9: Run Program

Name : Run Program

Actor(s):

- Collaborator
- Project Owner

Preconditions:

- The user must have a project created in one of the supported programming languages (e.g., C, C++, Java).
- The user must have access to the code editor and sidebar with input/output boxes.

Postconditions:

- The program is compiled, and either the output or any compilation errors are displayed in the output box.

Main Flow:

1. **User Action:** The user opens their project, which contains code in a specific programming language (e.g., C, C++, Java).
   System Response: Displays the code editor with the existing project code and a sidebar containing fields for input and output.

2. **User Action:** The user enters input into the input field in the sidebar (if applicable for the program being run).
   System Response: Displays the entered input in the input section of the sidebar.

3. **User Action:** The user clicks the "Run" button to compile and execute the code.
   System Response:
   - Compiles the code in the selected programming language (C, C++, Java, etc.).
   - If the code compiles successfully, the system executes the program and generates the output based on the entered input.
   - The output is displayed in the output box of the sidebar.
   - If there are any compilation errors, the system displays the error messages in the output box.

4. **User Action:** The user views the output or error messages.
   System Response: The output box shows the results of the executed program or any compilation errors.

Alternative Flows:

- **Compilation Errors:**
  1. The system detects errors during compilation (e.g., syntax errors, missing files).
  2. The system displays the error messages in the output box.

- **Invalid Input:**
    1. If the user enters invalid input (e.g., missing required fields), the system shows an error message: "Please enter valid input."

Scenario (User Action and System Response):

1. **User Action:** The user opens a Java project and enters some input in the sidebar's input field.
   System Response: Displays the entered input in the input box.

2. **User Action:** The user clicks the "Run" button to compile and execute the program.
   System Response: The system compiles the Java program, executes it, and displays the output or any errors in the output box on the sidebar.