TEEM

Unit Testing Document

Version 1

Testing Team: Mihir Paija

Dhruv Shah

Priyesh Tandel

Prepared On: 25-11-2023

INDEX

1. About TEEM

‘TEEM’ is a web app that aims to provide a platform for smoothing out the collaboration process between people from different backgrounds. It aims to do so by letting them create workspaces, organise meetings, and create tasks.

2. Objective

The objective of the unit testing is to ensure the accuracy and reliability of individual software units or components. This process involves verifying that each unit performs its intended function.

3. Technologies and Tools Used

The tech stack that has been used to develop the backend of the ‘TEEM’ web application is Node and Express. The code has been written in Typescript.

For the purpose of unit testing, we have used Jest along with Supertest. We chose Jest in conjunction with SuperTest for our unit testing needs due to their seamless integration and complementary strengths. Jest's native support for TypeScript, along with its efficient testing framework and features like snapshot testing and mocking, makes it a robust choice for unit testing our JavaScript and TypeScript codebase. SuperTest, specializing in HTTP assertions, seamlessly extends Jest's capabilities by enabling easy simulation of HTTP requests and assertions against our API endpoints. This combination provides a comprehensive testing suite, allowing us to validate both the individual units of our code and the functionality of our APIs within a unified and efficient testing environment. The expressive API of SuperTest and Jest's overall testing features create a powerful synergy that aligns well with our testing goals, ensuring thorough and reliable unit testing for our application.

4. Using Supertest with Jest

1. Authentication

1.1. Sign Up

API: /api/signup

Objective: To let the user signup

HTTP Method: POST

Developer: Krish Rupapara

TC(1.1.1):

Objective	Ensure the API returns a 400 status code when the email, name and password is missing.
Input	Empty Request Body
Expected Status Code	400
Expected Response	User signup should fail with an error message indicating that the Username and password is required.

TC(1.1.2):

Objective	Simulate an internal server error when invalid request body provided.
Input	User Data with only email id
Expected Status Code	500
Expected Response	An internal server error message should be returned.

TC(1.1.3):

Objective	Simulate an internal server error when invalid request body provided.

Input	User Data with only name
Expected Status Code	500
Expected Response	An internal server error message should be returned.

TC(1.1.4):

Objective	Simulate an internal server error when an invalid request body is provided.
Input	User Data with only password
Expected Status Code	500
Expected Response	An internal server error message should be returned.

TC(1.1.5):

Objective	Verify that the API returns a 200 status code and a success message when creating a User with required details.
Input	User Data with minimal required details
Expected Status Code	200
Expected Response	User Created successfully

1.2. Login

API: /api/login

Objective: To let the user login

HTTP Method: POST

Developer: Krish Rupapara and Mihir Paija

TC(1.2.1):

Objective	Ensure the API returns a 400 status code

	when the email and password are missing.
Input	Empty Request Body
Expected Status Code	400
Expected Response	User login should fail with an error message indicating that the email and password is required.

TC(1.2.2):

Objective	Simulate an internal server error when invalid request body provided.
Input	User Data with only email id
Expected Status Code	500
Expected Response	An internal server error message should be returned.

TC(1.2.3):

Objective	Simulate an internal server error when invalid request body provided.
Input	User Data with only password
Expected Status Code	500
Expected Response	An internal server error message should be returned.

TC(1.2.4):

Objective	Verify that the API returns a 200 status code and a success message when user login with required details.
Input	User Data with minimal required details
Expected Status Code	200
Expected Response	User Login successfully

1.3. Change Password

API: /api/changePassword

Objective: To let the user change Password when he knows password

HTTP Method: POST

TC(1.3.1):

Objective	Simulate an internal server error when invalid request body provided.
Input	Only oldPassword
Expected Status Code	400
Expected Response	change password should fail with an error message indicating that invalid/insufficient Password

TC(1.3.2):

Objective	Simulate an internal server error when invalid request body provided.
Input	Only newPassword
Expected Status Code	400
Expected Response	change password should fail with an error message indicating that invalid/insufficient Password

TC(1.3.3):

Objective	Simulate an internal server error when invalid request body provided.

Input	Only confirmPassword
Expected Status Code	400
Expected Response	User change password should fail with an error message indicating that invalid/insufficient Password

TC(1.3.4):

Objective	Return status code 400 when new password is same as old password
Input	oldPassword, newPassword and oldPassword.
Expected Status Code	400
Expected Response	Error message : old password cannot be same as new password.

TC(1.3.5):

Objective	Return status code 400 when new password is not same as confirm password
Input	oldPassword, newPassword and oldPassword.
Expected Status Code	400
Expected Response	Error message :confirm password must be same as new password.

TC(1.3.6):

Objective	Verify that the API returns a 200 status code and a success message when user’s password is changed.
Input	oldPassword, newPassword and oldPassword.
Expected Status Code	200
Expected Response	Success message : Change password successfully

1.4. Forgot Password

API: /api/forgotPassword

Objective: To let the user change when Password is forgotten

HTTP Method: POST

TC(1.4.1):

Objective	Ensure the API returns a 400 status code when the email id is missing.
Input	Empty Request Body
Expected Status Code	400
Expected Response	forgot password should fail with an error message indicating that Please provide email

TC(1.4.2):

Objective	Ensure the API returns a 200 when entered valid email id.
Input	Email id
Expected Status Code	200
Expected Response	forgot password should fail with an error message indicating that OTP sent successfully

TC(1.4.2):

Objective	Ensure the API returns a 400 when entered invalid email id.
Input	Email id
Expected Status Code	400

Expected Response	forgot password should fail with an error message indicating invalid credentials

1.4. Reset Password

API: /api/resetPassword

Objective: To let the user change when Password is forgotten

HTTP Method: POST

TC(1.4.1):

Objective	Ensure the API returns a 400 status code when the email or otp or password is missing.
Input	Empty email or otp or password
Expected Status Code	400
Expected Response	User reset password should fail with an error message indicating that Invalid, Insufficient email or OTP or Password

TC(1.4.2):

Objective	Ensure the API returns a 400 when OTP is expired
Input	Email , OTP , Password
Expected Status Code	200
Expected Response	User signup should fail with an error message indicating that OTP sent successfully

TC(1.4.3):

Objective	Ensure the API returns a 400 when entered

	invalid otp.
Input	Email , OTP , Password
Expected Status Code	400
Expected Response	User signup should fail with an error message indicating invalid otp

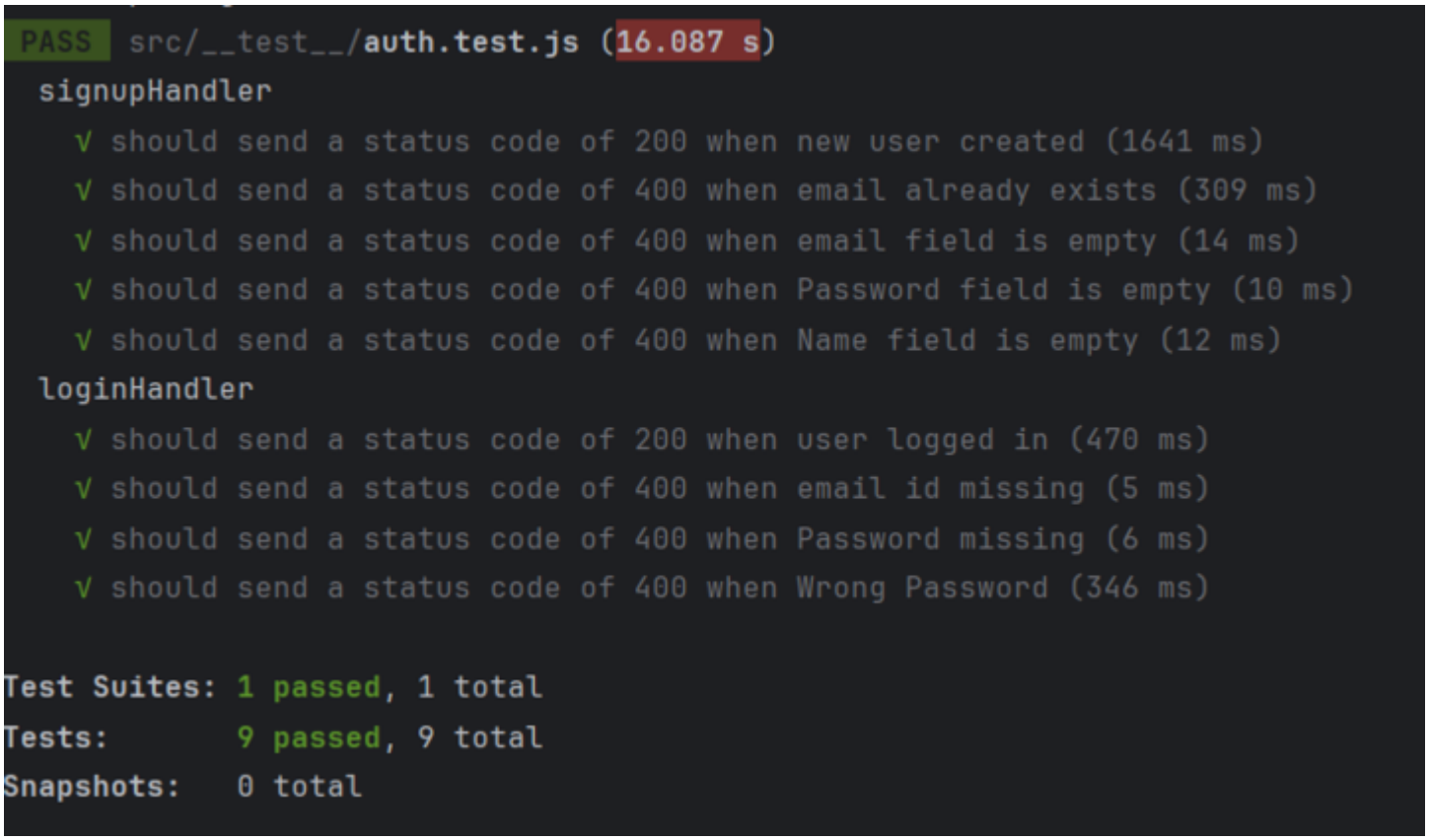
TC(1.4.4):

Objective	Ensure the API returns a 400 when entered invalid email.
Input	Email , OTP , Password
Expected Status Code	400
Expected Response	User signup should fail with an error message indicating invalid credentials.

TC(1.4.5):

Objective	Ensure the API returns a 400 when new password is same as current password.
Input	Email id
Expected Status Code	400
Expected Response	User signup should fail with an error message indicating new password is same current password.

Result:



2. Workspace Management

2.1. Create Workspace

API: /api/createWorkspace

Objective: The objective of the given API is to let the user create a workspace.

HTTP Method: POST

Middlewares Used: requireAuth

Developer: Mihir Paija

TC(2.1.1):

Objective	Ensure the API returns a 400 status code when the title is missing.
Input	Empty Request Body
Expected Status Code	400

Expected Response	Workspace creation should fail with an error message indicating that the title is required.

TC(2.1.2):

Objective	Simulate an internal server error when invalid request body provided.
Input	Workspace Data with only title
Expected Status Code	500
Expected Response	An internal server error message should be returned.

TC(2.1.3):

Objective	Verify that the API returns a 201 status code and a success message when creating a workspace with required details.
Input	Workspace Data with minimal required details
Expected Status Code	201
Expected Response	Workspace Created successfully

TC(2.1.4):

Objective	Ensure the API handles the creation of a workspace with unregistered members and returns the appropriate response.
Input	Workspace Data with unregistered member
Expected Status Code	201
Expected Response	Workspace Created without unregistered Members

TC(2.1.5):

Objective	Verify that the API returns a 201 status code and a success message when creating a

	workspace with registered members.
Input	Workspace Data with registered members
Expected Status Code	201
Expected Response	Workspace Created successfully.

2.1. Edit Workspace Details

API: /api/:wsID/editWSDetails

Objective: The objective of the given API is to let the workspace manager edit a workspace.

HTTP Method: GET

Middlewares Used: requireAuth,wsExist,authManager

Developer: Dhruv Shah

TC(2.2.1):

Objective	Verify that the API returns a 400 status code when the workspace ID is not a number.
Input	Workspace ID as a non-numeric value (e.g., "workspace_id")
Expected Status Code	400
Expected Response	Invalid wsID error message should be returned.

TC(2.2.2):

Objective	Ensure the API returns a 400 status code when the workspace ID is not passed.
Input	Empty Workspace ID
Expected Status Code	400
Expected Response	Invalid wsID error message should be returned.

TC(2.2.3):

Objective	Verify that the API returns a 404 status code when the specified workspace does not exist.
Input	Non-existing Workspace ID (e.g., 987)
Expected Status Code	404
Expected Response	Workspace Doesn't Exist error message should be returned.

TC(2.2.4):

Objective	Ensure the API returns a 401 status code when an unauthorized manager attempts to edit the workspace.
Input	Workspace ID with an unauthorized manager
Expected Status Code	401
Expected Response	You do not own the workspace error message should be returned.

TC(2.2.5):

Objective	Verify that the API returns a 200 status code and the updated workspace details when the workspace is edited successfully.
Input	Valid Workspace ID (e.g., 19)
Expected Status Code	200
Expected Response	workspace details should be returned.

HTTP Method: PATCH

Middlewares Used: requireAuth,wsExist,authManager

Developer: Dhruv Shah

TC(2.2.6):

Objective	Verify that the API returns a 400 status code when the workspace ID is not a number.
Input	Workspace ID as a non-numeric value (e.g., "workspace_id")
Expected Status Code	400
Expected Response	Invalid wsID error message should be returned.

TC(2.2.7):

Objective	Ensure the API returns a 400 status code when the workspace ID is not passed.
Input	Empty Workspace ID
Expected Status Code	400
Expected Response	Invalid wsID error message should be returned.

TC(2.2.8):

Objective	Verify that the API returns a 404 status code when the specified workspace does not exist.
Input	Non-existing Workspace ID (e.g., 987)
Expected Status Code	404
Expected Response	Workspace Doesn't Exist error message should be returned.

TC(2.2.9):

Objective	Ensure the API returns a 401 status code when an unauthorized manager attempts to edit the workspace.
Input	Workspace ID with an unauthorized manager
Expected Status Code	401
Expected Response	You do not own the workspace error message should be returned.

TC(2.2.10):

Objective	Ensure the API returns a 400 status code when the request body is empty.
Input	Empty Request Body
Expected Status Code	400
Expected Response	Error message indicating that Title, description, and type are required.

TC(2.2.11):

Objective	Verify that the API returns a 400 status code when the title is empty.
Input	Title, type, and description set to null
Expected Status Code	400
Expected Response	Error message indicating that Title can not be empty.

TC(2.2.12):

Objective	Ensure the API returns a 400 status code when the type is empty.
Input	Workspace data with type set to null
Expected Status Code	400
Expected Response	Error message indicating that Type can not be empty.

TC(2.2.13):

Objective	Verify that the API returns a 200 status code and a success message when the workspace is successfully edited with a null description
Input	Valid workspace data with null description
Expected Status Code	200

Expected Response	Success message indicating that Settings Saved.

TC(2.2.14):

Objective	Verify that the API returns a 200 status code and a success message when the workspace is successfully edited with complete workspace data.
Input	Valid workspace data with title, type, and description
Expected Status Code	200
Expected Response	Success message indicating that Settings Saved.

2.3. Edit Workspace Members

API: /api/:wsID/editWSMembers

Objective: The objective of the given API is to let the workspace manager edit a workspace members.

HTTP Method: GET

Middlewares Used: requireAuth,wsExist,authManager

Developer: Dhruv Shah

TC(2.3.1):

Objective	Verify that the API returns a 400 status code when the workspace ID is not a number.
Input	Workspace ID as a non-numeric value (e.g., "workspace_id")
Expected Status Code	400
Expected Response	Invalid wsID error message should be

	returned.

TC(2.3.2):

Objective	Ensure the API returns a 400 status code when the workspace ID is not passed.
Input	Empty Workspace ID
Expected Status Code	400
Expected Response	Invalid wsID error message should be returned.

TC(2.3.3):

Objective	Verify that the API returns a 404 status code when the specified workspace does not exist.
Input	Non-existing Workspace ID (e.g., 987)
Expected Status Code	404
Expected Response	Workspace Doesn't Exist error message should be returned.

TC(2.3.4):

Objective	Ensure the API returns a 401 status code when an unauthorized manager attempts to edit the workspace.
Input	Workspace ID with an unauthorized manager
Expected Status Code	401
Expected Response	You do not own the workspace error message should be returned.

TC(2.3.5):

Objective	Verify that the API returns a 200 status code and the updated workspace details when the workspace is edited successfully.

Input	Valid Workspace ID (e.g., 19)
Expected Status Code	200
Expected Response	workspace member list should be returned.

HTTP Method: PATCH

Middlewares Used: requireAuth,wsExist,authManager

Developer: Dhruv Shah

TC(2.3.6):

Objective	Verify that the API returns a 400 status code when the workspace ID is not a number.
Input	Workspace ID as a non-numeric value (e.g., "workspace_id")
Expected Status Code	400
Expected Response	Invalid wsID error message should be returned.

TC(2.3.7):

Objective	Ensure the API returns a 400 status code when the workspace ID is not passed.
Input	Empty Workspace ID
Expected Status Code	400
Expected Response	Invalid wsID error message should be returned.

TC(2.3.8):

Objective	Verify that the API returns a 404 status code when the specified workspace does not exist.
Input	Non-existing Workspace ID (e.g., 987)

Expected Status Code	404
Expected Response	Workspace Doesn't Exist error message should be returned.

TC(2.3.9):

Objective	Ensure the API returns a 401 status code when an unauthorized manager attempts to edit the workspace.
Input	Workspace ID with an unauthorized manager
Expected Status Code	401
Expected Response	You do not own the workspace error message should be returned.

TC(2.3.10):

Objective	Verify that the API returns a 201 status code and a success message when new members, including an unregistered member, are added to the workspace.
Input	Member data with new members, including an unregistered membe
Expected Status Code	201
Expected Response	Success message indicating that settings saved without unregistered members and they are invited to join TEEM.

TC(2.3.11):

Objective	Ensure the API returns a 200 status code and a success message when members are removed from the workspace.
Input	Member list without members to be removed
Expected Status Code	200
Expected Response	Success message indicating that Settings Saved.

TC(2.3.12):

Objective	Verify that the API returns a 200 status code and a success message when new members are added to the workspace.
Input	Member list with new (registered) members
Expected Status Code	200
Expected Response	Success message indicating that Settings Saved.

2.1. Delete Workspace

API: /api/:wsID/editWSDetails

Objective: The objective of the given API is to let the workspace manager edit a workspace members.

HTTP Method: DELETE

Middlewares Used: requireAuth,wsExist,authManager

Developer: Priyesh Tandel

TC(2.4.1):

Objective	Verify that the API returns a 400 status code when the workspace ID is not a number.
Input	Workspace ID as a non-numeric value (e.g., "workspace_id")
Expected Status Code	400
Expected Response	Invalid wsID error message should be returned.

TC(2.4.2):

Objective	Ensure the API returns a 400 status code when the workspace ID is not passed.
Input	Empty Workspace ID
Expected Status Code	400
Expected Response	Invalid wsID error message should be returned.

TC(2.4.3):

Objective	Verify that the API returns a 404 status code when the specified workspace does not exist.
Input	Non-existing Workspace ID (e.g., 987)
Expected Status Code	404
Expected Response	Workspace Doesn't Exist error message should be returned.

TC(2.4.4):

Objective	Ensure the API returns a 401 status code when an unauthorized manager attempts to edit the workspace.
Input	Workspace ID with an unauthorized manager
Expected Status Code	401
Expected Response	You do not own the workspace error message should be returned.

TC(2.4.5):

Objective	Verify that the API returns a 200 status code and a success message when a workspace is deleted.
Input	Valid Workspace ID (e.g., 22)
Expected Status Code	200
Expected Response	Success message indicating that Workspace deleted successfully.

Result:

```
PASS src/__test__/workspace.test.ts (24.621 s)
  createWorkspacePost
    √ should return 400 if title is missing (96 ms)
    √ should return 500 if an internal server error occurs (1219 ms)
    √ should return 201 with a success message if workspace is created successfully without details except titile (735 ms)
    √ should return 201 with a message and details if workspace is created with unregistered members (730 ms)
    √ should return 201 with a success message if workspace is created successfully (1576 ms)
  editWsDetailsGET
    √ should return 200 with a success message if workspace is edited successfully (352 ms)
    invlaid request body
      √ should return 400 with if the workspace_id is not a number (4 ms)
      √ should return 400 with if workspace_id is not passed (8 ms)
      √ should return 404 with if the workspace does not exist (176 ms)
    unauthorized manager
      √ should return 401 with if the user do not own workspace (177 ms)
  editWsDetailsPATCH
    invalid workspace ID
      √ should return 400 with if the workspace_id is not a number (5 ms)
      √ should return 400 with if workspace_id is not passed (8 ms)
      √ should return 404 with if the workspace does not exist (176 ms)
    unauthorized manager
      √ should return 401 with if the user do not own workspace (172 ms)
    invalid request body
      √ should return 400 with a success message if req body is empty (172 ms)
      √ should return 400 with a success message if title is empty (209 ms)
      √ should return 400 with a success message if type is empty (199 ms)
    valid request body
      √ should return 200 with a success message if workspace is edited successfully with null description (420 ms)
      √ should return 200 with a success message if workspace is edited successfully (362 ms)
  editWsMembersGET
    √ should return 200 with a success message if workspace is edited successfully (367 ms)
    invalid request body
      √ should return 400 with if the workspace_id is not a number (15 ms)
      √ should return 400 with if workspace_id is not passed (9 ms)
      √ should return 404 with if the workspace does not exist (190 ms)
    unauthorized manager
      √ should return 401 with if user do not own workspace (189 ms)
  editWsMembersPATCH
    invalid workspace ID
      √ should return 400 with if the workspace_id is not a number (16 ms)
      √ should return 400 with if workspace_id is not passed (12 ms)
      √ should return 404 with if the workspace does not exist (185 ms)
    unauthorized manager
      √ should return 401 with if user do not own workspace (193 ms)
    valid request body
      √ should return 201 with a success message if new member is unregister (2123 ms)
      √ should return 200 with a success message if workspace member removed (1383 ms)
      √ should return 200 with a success message if workspace member added (1789 ms)
  editWsWSDetailsDELETE
    √ should return 200 with if the workspace deleted (1365 ms)
    invalid workspace ID
      √ should return 400 with if the workspace_id is not a number (4 ms)
      √ should return 400 with if workspace_id is not passed (7 ms)
      √ should return 404 with if the workspace does not exist (169 ms)
    unauthorized manager
      √ should return 401 with if user do not own workspace (171 ms)

Test Suites: 1 passed, 1 total
Tests:       36 passed, 36 total
Snapshots:   0 total
Time:        24.809 s, estimated 30 s
Ran all test suites matching /workspace.test.ts/i.
```

3. Workspace Dashboard

3.1. Get Stream

API: /api/:wsID/stream

Objective: The objective of this API is to let the workspace member see the list of the all tasks and meets.

HTTP Method: GET

Middlewares Used: requireAuth,wsExist,authMember

Developer: Mihir Paija

TC(3.1.1):

Objective	Verify that the API returns a 400 status code when the workspace ID is not a number.
Input	Workspace ID as a non-numeric value (e.g., "workspace_id")
Expected Status Code	400
Expected Response	Invalid wsID error message should be returned.

TC(3.1.2):

Objective	Ensure the API returns a 400 status code when the workspace ID is not passed.
Input	Empty Workspace ID
Expected Status Code	400
Expected Response	Invalid wsID error message should be returned.

TC(3.1.3):

Objective	Verify that the API returns a 404 status code when the specified workspace does not exist.
Input	Non-existing Workspace ID (e.g., 987)

Expected Status Code	404
Expected Response	Workspace Doesn't Exist error message should be returned.

TC(3.1.4):

Objective	Ensure the API returns a 401 status code when an unauthorized member attempts to view stream of the workspace.
Input	Workspace ID with an unauthorized member
Expected Status Code	401
Expected Response	You are not part of this workspace error message should be returned.

TC(3.1.5):

Objective	Verify that the API returns a 200 status code and the stream data for the workspace manager
Input	Valid Workspace ID (e.g., 19)
Expected Status Code	200
Expected Response	List of list of the all tasks and meets of workspace should be returned.

TC(3.1.6):

Objective	Verify that the API returns a 200 status code and the stream data for the workspace member
Input	Valid Workspace ID (e.g., 19)
Expected Status Code	200
Expected Response	List of list of the all tasks and meets of workspace should be returned.

3.2. Get People

API: /api/:wsID/people

Objective: The objective of this API is to let the workspace member see the list of people involved in the workspace.

HTTP Method: GET

Middlewares Used: requireAuth,wsExist,authMember

Developer: Mihir Paija

TC(3.2.1):

Objective	Verify that the API returns a 400 status code when the workspace ID is not a number.
Input	Workspace ID as a non-numeric value (e.g., "workspace_id")
Expected Status Code	400
Expected Response	Invalid wsID error message should be returned.

TC(3.2.2):

Objective	Ensure the API returns a 400 status code when the workspace ID is not passed.
Input	Empty Workspace ID
Expected Status Code	400
Expected Response	Invalid wsID error message should be returned.

TC(3.2.3):

Objective	Verify that the API returns a 404 status code when the specified workspace does not exist.
Input	Non-existing Workspace ID (e.g., 987)
Expected Status Code	404

Expected Response	Workspace Doesn't Exist error message should be returned.

TC(3.2.4):

Objective	Ensure the API returns a 401 status code when an unauthorized member attempts to view the list of people of the workspace.
Input	Workspace ID with an unauthorized member
Expected Status Code	401
Expected Response	You are not part of this workspace error message should be returned.

TC(3.2.5):

Objective	Verify that the API returns a 200 status code and the list of people involved in workspace.
Input	Valid Workspace ID (e.g., 19)
Expected Status Code	200
Expected Response	List of people involved in workspace should be returned.

3.3. Get YourWork

API: /api/:wsID/yourWork?filter=

Objective: The objective of this API is to let the workspace member see the list of tasks that are assigned to them in the workspace using filter option like “upcoming” and “all”.

HTTP Method: GET

Middlewares Used: requireAuth,wsExist,authMember

Developer: Mihir Paija

TC(3.3.1):

Objective	Verify that the API returns a 400 status code when the workspace ID is not a number.
Input	Workspace ID as a non-numeric value (e.g., "workspace_id")
Expected Status Code	400
Expected Response	Invalid wsID error message should be returned.

TC(3.3.2):

Objective	Ensure the API returns a 400 status code when the workspace ID is not passed.
Input	Empty Workspace ID
Expected Status Code	400
Expected Response	Invalid wsID error message should be returned.

TC(3.3.3):

Objective	Verify that the API returns a 404 status code when the specified workspace does not exist.
Input	Non-existing Workspace ID (e.g., 987)
Expected Status Code	404
Expected Response	Workspace Doesn't Exist error message should be returned.

TC(3.3.4):

Objective	Ensure the API returns a 401 status code when an unauthorized member attempts to view list their work in the workspace.
Input	Workspace ID with an unauthorized member

Expected Status Code	401
Expected Response	You are not part of this workspace error message should be returned.

TC(3.2.5):

Objective	Verify that the API returns a 200 status code and upcoming tasks for the workspace manager when the filter is set to "Upcoming".
Input	Valid Workspace ID (e.g., 19) Filter: Upcoming
Expected Status Code	200
Expected Response	List of all upcoming tasks of workspace should be returned.

TC(3.2.6):

Objective	Verify that the API returns a 200 status code and all tasks for the workspace manager when the filter is set to "All".
Input	Valid Workspace ID (e.g., 19) Filter: All
Expected Status Code	200
Expected Response	List of all tasks of workspace should be returned.

TC(3.2.7):

Objective	Verify that the API returns a 200 status code and upcoming tasks for the workspace member when the filter is set to "Upcoming".
Input	Valid Workspace ID (e.g., 24) Filter: Upcoming
Expected Status Code	200
Expected Response	List of upcoming tasks of workspace which are assigned to that member should be

	returned.

TC(3.2.8):

Objective	Verify that the API returns a 200 status code and all tasks for the workspace member when the filter is set to "All".
Input	Valid Workspace ID (e.g., 24) Filter: All
Expected Status Code	200
Expected Response	List of all tasks of workspace which are assigned to that member should be returned.

3.4. Get YourMeet

API: /api/:wsID/yourMeet?filter=

Objective: The objective of this API is to let the workspace member see the list of meets of the workspace in which they are invited using filter option like “upcoming” and “all”.

HTTP Method: GET

Middlewares Used: requireAuth,wsExist,authMember

Developer: Mihir Paija

Result:

```
PASS src/__test__/wsDashboard.test.ts (26.068 s)
  people
    √ should return 200 with if all good (840 ms)
    invalid workspace ID
      √ should return 400 with if the workspace_id is not a number (176 ms)
      √ should return 400 with if workspace_id is not passed (13 ms)
      √ should return 404 with if the workspace does not exist (926 ms)
    unauthorized member
      √ should return 401 with if the workspace does not exist (744 ms)
  Your Work
    invalid workspace ID
      √ should return 400 with if the workspace_id is not a number (9 ms)
      √ should return 400 with if workspace_id is not passed (8 ms)
      √ should return 404 with if the workspace does not exist (173 ms)
    unauthorized member
      √ should return 401 with if the workspace does not exist (402 ms)
    user is workspace manager
      √ should return 200 with if the workspace manager and filter is upcoming (339 ms)
      √ should return 200 with if theworkspace manager and filter is all (345 ms)
    user is workspace member
      √ should return 200 with if the workspace manager and filter is upcoming (592 ms)
      √ should return 200 with if theworkspace manager and filter is all (514 ms)
      √ should return 200 with if theworkspace manager and filter is not passed (633 ms)
  stream
    invalid workspace ID
      √ should return 400 with if the workspace_id is not a number (7 ms)
      √ should return 400 with if workspace_id is not passed (11 ms)
      √ should return 404 with if the workspace does not exist (165 ms)
    unauthorized member
      √ should return 401 with if the workspace does not exist (348 ms)
    user is workspace manager
      √ should return 200 with if the workspace manager (565 ms)
    user is workspace member
      √ should return 200 with if the workspace member (722 ms)

Test Suites: 1 passed, 1 total
Tests:       20 passed, 20 total
Snapshots:   0 total
Time:        26.307 s
Ran all test suites matching /wsDashboard.test.ts/i.
```

4. Task Testing

4.1 Assign Task

API: /api/:wsID/assignTask

Objective: The objective of this API is to let the project manager create a task within the workspace.

HTTP Method: POST

Middlewares Used: requireAuth, wsExist, authorizeManager

Developer: Mihir Paija

TC(4.1.1):

Objective	Verify that the API returns a 201 status code when all the details are provided and the assignees list contains all workspace members
Input	Task Details with Assignees List based on above condition
Expected Status Code	201
Expected Response	Task Creation should be successful and assigned to the given assignees.

TC(4.1.2):

Objective	Verify that the API returns a 201 status code when all the details are provided and the assignees list contains workspace as well as non workspace members
Input	Task Details with Assignees List based on above condition
Expected Status Code	201
Expected Response	Task Creation should be successful and assigned to only workspace members will a classification to who amongst the assignee is assigned and who isn't a part of the workspace.

TC(4.1.3):

Objective	Verify that the API returns a 201 status code when all the details are provided and the assignees list contains workspace as well as non registered members

Input	Task Details with Assignees List based on above condition
Expected Status Code	201
Expected Response	Task Creation should be successful and assigned to only workspace members will a classification to who amongst the assignee is assigned and who isn't a part of TEEM

TC(4.1.4):

Objective	Verify that the API returns a 201 status code when all the details are provided and the assignees list contains workspace as well as non registered members
Input	Task Details with Assignees List based on above condition
Expected Status Code	201
Expected Response	Task Creation should be successful and assigned to only workspace members will a classification to who amongst the assignee is assigned and who isn't a part of the workspace and who isn’t a part of TEEM

TC(4.1.5):

Objective	Verify that the API returns a 401 status code when the user creating the task is not the project manager
Input	Task Details with Assignees List
Expected Status Code	401
Expected Response	Task Creation should fail with an error message that the user isn’t the project manager to the given workspace.

TC(4.1.6):

Objective	Verify that the API returns a 400 status code when the title is missing
Input	Task Details with Assignees List
Expected Status Code	400
Expected Response	Task Creation should fail with an error message that the title is missing

TC(4.1.7):

Objective	Verify that the API returns a 400 status code when the deadline is missing
Input	Task Details with Assignees List
Expected Status Code	400
Expected Response	Task Creation should fail with an error message that the deadline is missing

TC(4.1.8):

Objective	Verify that the API returns a 404 status code when the given workspace doesn’t exist
Input	Task Details with Assignees List
Expected Status Code	404
Expected Response	Task Creation should fail with an error message that the workspace doesn’t exist

TC(4.1.9):

Objective	Verify that the API returns a 404 status code when an invalid workspace ID is sent
Input	Task Details with Assignees List
Expected Status Code	404

Expected Response	Task Creation should fail with an error message that the workspace ID is inavlid

4.2 Get Task Details

API: /api/:wsID/:taskID/editTaskDetails

Objective: The objective of this API is to let the project manager fetch the task details so he/she could then edit it.

HTTP Method: GET

Middlewares Used: requireAuth, wsExist, authorizeManager, taskExist, getTaskDetails

Developer: Dhruv Shah

TC(4.2.1):

Objective	Verify that the API returns a 200 status code when the project manager tries to fetch task details
Input	It is a GET Request
Expected Status Code	200
Expected Response	The task details should be displayed.

TC(4.2.2):

Objective	Verify that the API returns a 404 status code when the workspace provided doesn’t exist
Input	It is a GET Request
Expected Status Code	404
Expected Response	The fetch should not be successful with an error message that the workspace doesn’t exist

TC(4.2.3):

Objective	Verify that the API returns a 404 status code when the given task doesn’t exist within the given workspace
Input	It is a GET Request
Expected Status Code	404
Expected Response	The fetch should not be successful with an error message that the given task doesn’t exist in the given workspace.

TC(4.2.4):

Objective	Verify that the API returns a 400 status code when an invalid workspace ID is provided
Input	It is a GET Request
Expected Status Code	400
Expected Response	The fetch should not be successful with an error message that the given workspace ID is invalid.

TC(4.2.5):

Objective	Verify that the API returns a 400 status code when an invalid task ID is provided
Input	It is a GET Request

Expected Status Code	400
Expected Response	The fetch should not be successful with an error message that the given task ID is invalid.

4.3 Edit Task Details

API: /api/:wsID/:taskID/editTaskDetails

Objective: The objective of this API is to let the project manager edit the task details of an already created task.

HTTP Method: PATCH

Middlewares Used: requireAuth, wsExist, authorizeManager, taskExist, getTaskDetails

Developer: Dhruv Shah

TC(4.3.1):

Objective	Verify that the API returns a 200 status code when everything is correctly provided
Input	Updated Task Details
Expected Status Code	200
Expected Response	Task Updation should be successful

TC(4.3.2):

Objective	Verify that the API returns a 400 status code when the title isn’t provided

Input	Updated Task Details without title
Expected Status Code	400
Expected Response	Task Updation should fail with an message that necessary fields are missing

TC(4.3.3):

Objective	Verify that the API returns a 400 status code when the description isn’t provided
Input	Updated Task Details without description
Expected Status Code	400
Expected Response	Task Updation should fail with an message that necessary fields are missing.

TC(4.3.4):

Objective	Verify that the API returns a 400 status code when the type isn’t provided
Input	Updated Task Details without type
Expected Status Code	400
Expected Response	Task Updation should fail with an message that necessary fields are missing.

TC(4.3.5):

Objective	Verify that the API returns a 400 status code when the status isn’t provided
Input	Updated Task Details without status
Expected Status Code	400
Expected Response	Task Updation should fail with an message that necessary fields are missing

TC(4.3.6):

Objective	Verify that the API returns a 400 status code when the title provided is empty
Input	Updated Task Details with empty title
Expected Status Code	400
Expected Response	Task Updation should fail with an message that the title can’t be empty

TC(4.3.7):

Objective	Verify that the API returns a 400 status code when an invalid workspace ID is provided
Input	Updated Task Details
Expected Status Code	400
Expected Response	Task Updation should fail with an error message that the given workspace ID is invalid.

TC(4.3.8):

Objective	Verify that the API returns a 400 status code when an invalid task ID is provided
Input	Updated Task Details
Expected Status Code	400
Expected Response	Task Updation should fail with an error message that the given task ID is invalid.

4.4 Get Task Assignees

API: /api/:wsID/:taskID/editTaskAssignees

Objective: The objective of this API is to let the project manager fetch the task assignees so he/she could then edit it.

HTTP Method: GET

Middlewares Used: requireAuth, wsExist, authorizeManager, taskExist, getTaskDetails

Developer: Dhruv Shah

TC(4.4.1):

Objective	Verify that the API returns a 200 status code when a project manager asks for the assignee list
Input	Its a GET Request
Expected Status Code	200
Expected Response	The assignee list should be successfully displayed.

4.5 Edit Task Assignees

API: /api/:wsID/:taskID/editTaskAssignees

Objective: The objective of this API is to let the project manager edit the task assignees of an already created task.

HTTP Method: PATCH

Middlewares Used: requireAuth, wsExist, authorizeManager, taskExist, getTaskDetails

Developer: Dhruv Shah

TC(4.5.1):

Objective	Verify that the API returns a 201 status code when a project manager updates a assignee list

Input	A non empty assignee list with all workspace members
Expected Status Code	201
Expected Response	The task assignees should be successfully updated.

TC(4.5.2):

Objective	Verify that the API returns a 201 status code when a project manager updates a assignee list
Input	A non empty assignee list with assignees that are part of workspace, not part of workspace and not part of TEEM
Expected Status Code	201
Expected Response	The task assignees should be successfully updated and only assigned to workspace members with the assignees being classified into categories mentioned in the input.

TC(4.5.3):

Objective	Verify that the API returns a 400 status code if the assignee list is empty
Input	Empty assignee list
Expected Status Code	400
Expected Response	Task Assignees Updation should fail with an error message that the assignee list is empty

TC(4.5.4):

Objective	Verify that the API returns a 400 status code if the assignee list contains no workspace members
Input	Assignee list with no workspace members
Expected	400

Status Code	
Expected Response	Task Assignees Updation should fail with an error message that the assignee list has no workspace members

4.6 Delete Task

API: /api/:wsID/:taskID/editTaskDetails

Objective: The objective of this API is to let the project manager delete an already created task.

HTTP Method: DELETE

Middlewares Used: requireAuth, wsExist, authorizeManager, taskExist

Developer: Priyesh Tandel

TC(4.6.1):

Objective	Verify that the API returns a 200 status code if the project manager makes a delete request
Input	It's a delete request
Expected Status Code	200
Expected Response	Task should be deleted successfully.

TC(4.6.2):

Objective	Verify that the API returns a 401 status code a non project manager makes a delete request
Input	It's a delete request

Expected Status Code	401
Expected Response	Task Deletion should fail with an error message that the user isn't the project manager of the given workspace.

4.7 Task Dashboard

API: /api/:wsID/:taskID/taskDashboard

Objective: The objective of this API is to let the assignees of the task and the project manager see the details and assignees of the task.

HTTP Method: GET

Middlewares Used: requireAuth, wsExist, authorizeMember, taskExist, authorizeAssignee

Developer: Mihir Paija

TC(4.7.1):

Objective	Verify that the API returns a 200 status code when a assignee wants to see a task
Input	It is a GET Request
Expected Status Code	200
Expected Response	The details of the task along with the assignee list should be displayed

TC(4.7.2):

Objective	Verify that the API returns a 401 status code when a non-assignee tries to fetch details
Input	It is a GET Request

Expected Status Code	401
Expected Response	The fetch should not be successful with an error message that the user has not been assigned the given task.

Results:

```
PASS src/__test__/task.test.ts (40.38 s)
  assignTaskPost
    √ should assign a task with all workspace assignees and return a status code of 201 with success message (2152 ms)
    √ should assign a task only to workspace members and return a status code of 201 with success message and classify non workspace but registered assignees (1584 ms)

    √ should assign a task only to workspace members and return a status code of 201 with success message and classify non-registered assignees (1050 ms)

    √ should assign a task only to workspace members and return a status code of 201 with success message and classify other assignees (1928 ms)
    √ should return a status code of 401 and an error message that the user is not the owner of the workspace (175 ms)
    √ should return a status code of 400 and an error message for missing title (167 ms)
    √ should return a status code of 400 and an error message for missing deadline (169 ms)
    √ should return a status code of 404 as the workspace doesnt exist (162 ms)
    √ should return a status code of 400 and an error message of invalid workspace ID (7 ms)
  editTaskDetailsGet
    √ should fetch the task details successfully and return a status code of 200 (724 ms)
    √ should return a status code of 404 as workspace doesnt exist (167 ms)
    √ should return a status code of 404 as task doesnt exist within the given workspace (398 ms)
    √ should return a status code of 400 and an error message of invalid workspace ID (5 ms)
    √ should return a status code of 400 and an error message of invalid task ID (181 ms)
  editTaskDetailsPATCH
    √ should edit task details successfully and return a status code of 200 (910 ms)
    √ should give an error as title is undefined (702 ms)
    √ should give an error as description is undefined (738 ms)
    √ should give an error as type is undefined (691 ms)
    √ should give an error as status is undefined (692 ms)
    √ should give an error as title is null (707 ms)
    √ should return a status code of 400 and an error message of invalid workspace ID (5 ms)
    √ should return a status code of 400 and an error message of invalid task ID (164 ms)
  editTaskAssigneesGet
    √ should fetch the task assignees successfully and return a status code of 200 (705 ms)
  editTaskAssigneesPATCH
    √ should edit task assignees successfully and return a status code 201 (1609 ms)
    √ should edit task assignees successfully to workspace members, return a status code 201 and classify other assignees (2123 ms)
    √ should handle empty Assignees (697 ms)
    √ should not make any changes if no assignee is part of the workspace (1029 ms)
  deleteTask
    √ should delete task succesfully and return a status code of 200 (703 ms)
    √ should return a status code of 401 with an error message that the user isnt the project manager (162 ms)
    √ should fetch the task details successfully and return a status code of 200 (689 ms)
    √ should return a status code of 401 as the user is not an assignee (681 ms)

Test Suites: 1 passed, 1 total
Tests:       31 passed, 31 total
Snapshots:   0 total
Time:        40.624 s
Ran all test suites matching /task.test.ts/i.
```

5. Dashboard Testing

5.1 Get Dashboard

API: /api/dashboard

Objective: The objective of this API is to let the user see the list of all workspaces that they are part of.

HTTP Method: GET

Middlewares Used: requireAuth

Developer: Mihir Paija

TC(5.1.1):

Objective	should return 200 with when user is valid and workspaces are there.
Input	Email , password
Expected Status Code	200
Expected Response	Workspace [] array.

5.2 Get Profile

API: /api/profile

Objective: The objective of this API is to let the user see his profile

HTTP Method: GET

Middlewares Used: requireAuth

Developer: Mihir Paija

TC(5.2.1):

Objective	should return 200 with when user is valid.
Input	Email , password
Expected Status Code	200
Expected Response	User Body.

TC(5.2.2):

Objective	should return 500 with when user is invalid.
Input	Email , password
Expected Status Code	500
Expected Response	Undefined

5.3 Update Profile

API: /api/profile

Objective: The objective of this API is to let the user update his profile.

HTTP Method: PATCH

Middlewares Used: requireAuth

Developer: Mihir Paija

TC(5.3.1):

Objective	should return 200 When all fields are provided properly
Input	User Body
Expected Status Code	200
Expected Response	Message : “User profile Updated”

TC(5.3.2):

Objective	should return 400 with when user is invalid.
Input	User Body
Expected Status Code	400
Expected Response	User not found.

TC(5.3.3):

Objective	should return 400 with when user is attempting to update email.
Input	User Body
Expected Status Code	400
Expected Response	You cannot change email ID.

TC(5.3.4):

Objective	should return 400 with when Username is missing
Input	User Body with empty Username field
Expected Status Code	400

Expected Response	Username cannot be empty

TC(5.3.5):

Objective	should return 400 with when email is missing
Input	User Body with empty emailID field
Expected Status Code	400
Expected Response	emailID cannot be empty

TC(5.2.6):

Objective	should return 400 with when email is missing
Input	Empty Body
Expected Status Code	400
Expected Response	Insufficient request body

5.4 Delete Profile

API: /api/profile

Objective: The objective of this API is to let the user delete his profile.

HTTP Method: DELETE

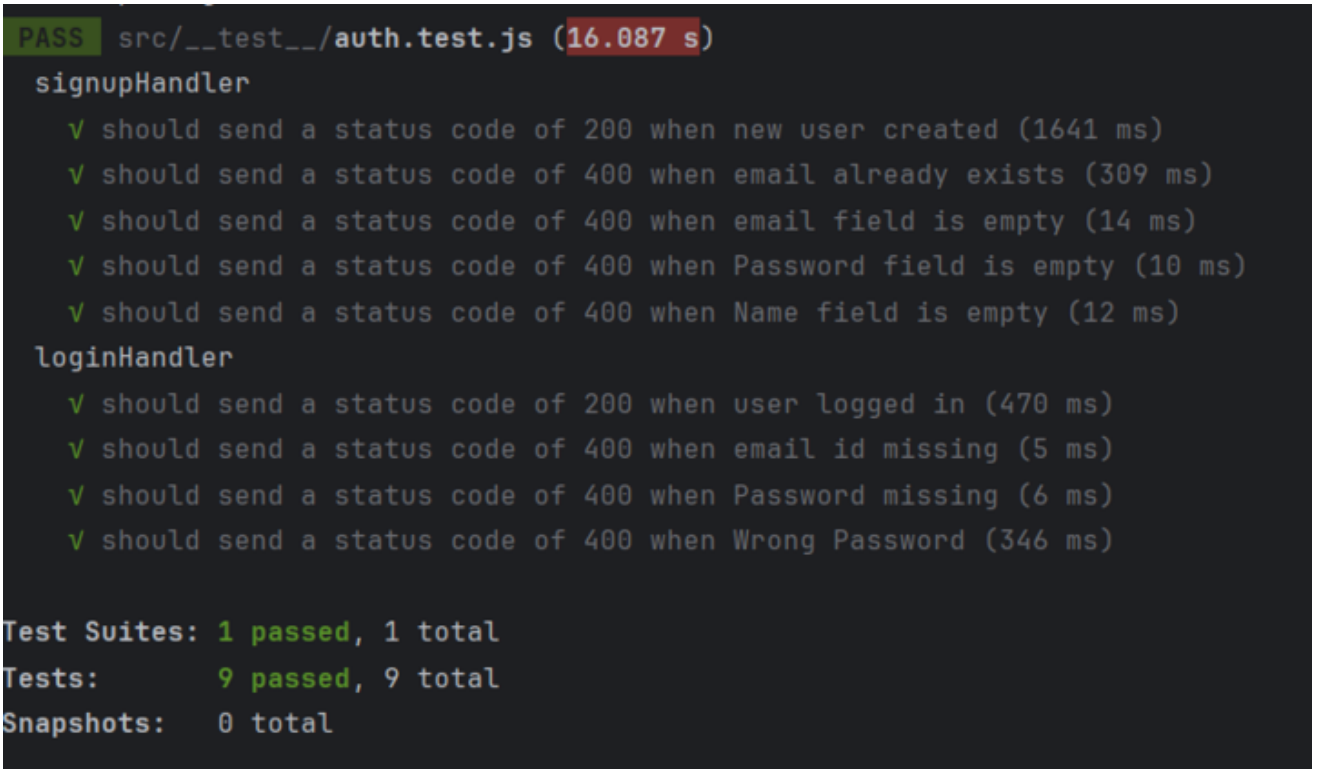
Middlewares Used: requireAuth

Developer: Priyesh Tandel

TC(5.4.1):

Objective	should return 200 when Profile deleted successfully.
Input	Email password from requireAuth
Expected Status Code	200
Expected Response	Message : “User profile Deleted Successfully.”

Result:



6. Meet Testing

6.1 Schedule Meet

API: /api/:wsid/scheduleMeet

Objective: The objective of this API is to let the user Schedule the meeting

HTTP Method: GET

Middlewares Used: requireAuth , wsExist

Developer: Krish Rupapara

TC(6.1.1):

Objective	should return 200 when the meeting is successfully scheduled
Input	summary , description, agenda, startDate, startTime , endTime, venue, participants: [ ]
Expected Status Code	200
Expected Response	Meet Successfully created

TC(6.1.2):

Objective	should return 400 when Title , agenda or date is empty
Input	Whole meetDetail object without any of the one above
Expected Status Code	400
Expected Response	Insufficient Fields

6.1 Edit Meet

API: /api/:wsid/scheduleMeet

Objective: The objective of this API is to let the user Edit the meeting

HTTP Method: GET

Middlewares Used: requireAuth , wsExist

Developer: Krish Rupapara

TC(6.2.1):

Objective	should return 200 when the meeting is successfully Edited
Input	summary , description, agenda, startDate, startTime , endTime, venue, participants: [ ]
Expected Status Code	200
Expected Response	Meet Successfully Edited

TC(6.2.2):

Objective	should return 400 when any of the field is missing
Input	Whole meetDetail object without any of the one above from summary , description, agenda, startDate, startTime , endTime, venue empty
Expected Status Code	400
Expected Response	Insufficient Fields

5. Conclusion

In conclusion, the unit testing process has been instrumental in ensuring the robustness and reliability of our codebase. Through comprehensive test coverage, we have tried our best to identify and address potential issues, ensuring that our application functions as intended across various scenarios. Successful test cases have provided confidence in the stability of our code and any challenges encountered were effectively mitigated.