**“Project Management Tool”**

**Technical Document**

Document Created

24/08/2021

Document Modified

-

Created By

Sabih Waseem Khan****

# INTRODUCTION

This document contains the technical walkthrough of the project’s backend. The main features of the project are listed below.

* Authorization
* Team Management
* Project Management
* Challenge Management
* Card Management

# Authorization

This module includes login, logout, signup, account verification etc. The system-level permissions (e.g. side menu)

**\***Authcontroller is the only controller which acts as user endpoint as well as contains the business logic of tis methods.

## Login/Logout

Registered user can login with their email and password combination or by using their google of facebook account. If an email is already linked with a non-social account, it can no longer be used to login with provider (google/facebook).

**Associated Database Tables**

* User
* Refreshtoken

**Associated Files**

* AuthController.cs
* LoginReq.cs
* SignupReq.cs
* SocialLoginReq.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [AuthController]  Login(LoginReq) | Authenticate the user logging in by verifying their email and password. |
| [AuthController]  SocialLogin(SocialLoginReq) | Loggs in the user who logged in by using the “Continue With Google” or “Continue With Facbook” functionality. If the user’s email does not exist in the database, it creates a new record for it. If the email is already used to singup a user without social feature then it can’t be used for social login and vice-versa. |
| [AuthController]  SignUp(SignUpReq) | Creates a new record for the user attempting to sign up. Also sends a verification email to the user to verify their account. If the email is already used with social feature then it cannot be used to signup a new user and vice-versa. |

## Reset/Verification

Account verification, resending token, resetting password and similar sub-features under this summarised under this feature.

**Associated Database Tables**

* User

**Associated Files**

* AuthController.cs
* ResetPasswordReq.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| ForgetPassword(string) | Creates a reset token for user, stores in database and emails it to user. |
| VerifyResetCode(string, string) | Confirms the reset code entered by the user with the code for the user in database. |
| ResendVerificationCode(string) | Resend the email verification code to the user. |
| AccountVerification(string, string) | Mark the user account as verified if the code entered by the user matches the code in the database for the user. |
| ChangePassword(ResetPasswordReq) | Update password for the current authorized user. |

# Team Management

This feature includes creation, deletion, retrieval and modification of teams.

## Team List

A user can view the teams created by himself/herself and the teams he/she is added in by other users. Also a team can be modified only by the team owner i.e. the person who has created the team.

**Associated Database Tables**

* Team
* TeamMember
* TemporaryTeamMember
* User

**Associated Files**

* TeamController.cs
* TeamService.cs
* GetAllPageReq.cs
* TeamListDto.cs
* TeamListDtoPage.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [TeamController]  GetAllTeams(GetAllPageReq) | Api endpoint exposed to the frontend which calls the service to retrive the list of teams |
| [TeamService]  GetAllTeams(GetAllPageReq, int?) | Gets the list of teams filtered according to the provided filterstring, page size and page count. |

## Team Create/Edit

A user can create a team by entering name, description. Also along with current registered members in the system, the members can be invited to join by their email.

**Associated Database Tables**

* Team
* TeamMember
* TemporaryTeamMember

**Associated Files**

* TeamController.cs
* TeamService.cs
* UpsertTeamReq.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [TeamController]  UpsertTeam(UpsertTeamReq) | Api endpoint exposed to the frontend which calls the service either insert or update a team. |
| [TeamService]  UpsertTeam(UpsertTeamReq,int?) | Create team (if not exist) and add its members, else update team. |

## Team Delete

A user can only delete the team created by himself/herself.

**Associated Database Tables**

* Team

**Associated Files**

* TeamController.cs
* TeamService.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [TeamController]  DeleteTeam(string) | Api endpoint exposed to the frontend which calls the service to soft-delete a team. |
| [TeamService]  DeleteTeam(string, int?) | Soft delete the team i.e. set isActive to false. |

# Project Management

This feature includes listing projects, create a project, edit a project, deleting a project, add or remove members etc.

## Project List

A user can view the projects created by himself/herself and the projects he/she is added in by other users. Also a project can be modified only by the project owner i.e. the person who has created the project.

**Associated Database Tables**

* Project
* ProjectTeam
* TeamMember
* ProjectMember
* TemporaryProjectMember
* Challenge
* Card

**Associated Files**

* ProjectController.cs
* ProjectService.cs
* GetAllPageReq.cs
* ProjectListDtoPage.cs
* ProjectListDto.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [ProjectController] GetProjects(GetAllPageReq) | Api endpoint exposed to the frontend which calls the service to retrieve the list of projects. |
| [ProjectService] GetProjects(GetAllPageReq, int?) | Retrieves all the projects of the current logged in user according to specified search text, page index and page count. |
| [ProjectService] IsProjectOwner(ProjectListDto, List<Project>, int?) | Determine if the current user is the owner of the project specified by ‘ProjectListDto’ |
| [ProjectService] GetTotalTasksCount(int) | Get the total count of the tasks in the specified project. |

## Project Create/Edit

A user can create a project by adding a project by entering name, due date etc. Also along with individual members, teams(created by the user itself) can be added in the project at once.

**Associated Database Tables**

* Project
* ProjectTeam
* ProjectMember
* TemporaryProjectMember
* Team
* TeamMember
* TemporaryTeamMember
* User
* Challenge
* ChallengeList
* ProjectMemberPermission

**Associated Files**

* ProjectController.cs
* CommonController.cs
* ProjectService.cs
* CreateProjectReq.cs
* EditProjectDto.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [ProjectController]  CreateProject(CreateProjectReq) | Api endpoint exposed to the frontend which calls the service to create a new record for a project. |
| [ProjectController] GetProjectByById(string) | Api endpoint exposed to the frontend which calls the service to retrive a projectby its public id. |
| [ProjectController] GetExistingProjectMembers(string) | Api endpoint exposed to the frontend which calls the service to get members of a project. |
| [CommonController] GetMemberDDL(string=null) | Api endpoint exposed to the frontend which calls the service to get members according to the filter query. |
| [CommonController]  GetTeamDDL() | Api endpoint exposed to the frontend which calls the service to get teams of a user. |
| [ProjectService] GetProjectById(string, int?) | Gets a project along with its members by its publicId for edit. |
| [ProjectService]  GetExistingProjectMembers(string) | Gets members already in the project to bind initial values in the dropdown on edit page. |
| [CommonService]  GetMemberDDL(string, int? | Gets all active members for the dropdown filtered by the string query provided. |
| [CommonService]  GetTeamDDL() | Gets all the teams created by the current logged-in user for dropdown. |
| [ProjectService]  CreateProject(CreateProjectReq, int?) | Retrieves all the projects of the current logged in user according to specified search text, page index and page count. |
| [ProjectService]  AddProjectTeam(string, int?, int) | Add selected teams in the project. |
| [Service] AddRegisteredMember(User, int?, Project) | Add members to the project that were selected on the frontend and already exist in the system’s database. |
| [ProjectService] AddUnregisteredMember(string, int?, Project) | Add the members to the project that were added on the frontend by their email and donot currently exist in the database. |
| [ProjectService] AddAdminPermissions(int, int) | Add all the default permissions for the project’s owner so he can access everything in the project. |
| [ProjectService] CreateChallengeTemplatesForProject(int, int, int?, int) | Add challenge records in the database for the specified challege count. |

## Project Delete

A user can only delete the projects created by himself/herself.

**Associated Database Tables**

* Project

**Associated Files**

* ProjectController.cs
* ProjectService.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [ProjectController]  DeleteProject(string) | Api endpoint exposed to the frontend which calls the service to delete a project. |
| [ProjectService]  DeleteProject(string, int?) | Soft delete the project i.e. set isActive to false. |

## Project Permissions

A project owner can grant permissions to a project member from a list of permissions. By default during the project creation the owner is granted all the permissions which cannot be changed. The owner can grant all the permissions to a member or no permissoins at all.

**Associated Database Tables**

* Project
* ProjectTeam
* TeamMember
* ProjectMember
* TemporaryProjectMember
* ProjectPermission
* ProjectMemberPermission

**Associated Files**

* ProjectController.cs
* ProjectService.cs
* GetAllPageReq.cs
* ProjectMemberPermissionDto.cs
* ProjectPermissionDto.cs
* ProjectParticipants.cs
* UpdateProjectPermissionRequest.cs
* ProjectListDtoPage.cs
* ProjectListDto.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [ProjectController] GetProjectMembers(string) | Api endpoint exposed to the frontend which calls the service to get all project members. |
| [ProjectController]  GetAllPermissions() | Api endpoint exposed to the frontend which calls the service to get list of all permissions. |
| [ProjectController] GetAllMemberPermissions(string) | Api endpoint exposed to the frontend which calls the service to get all the permissions that the members already have. |
| [ProjectController] UpdateProjectPermissions(string, List<UpdateProjectPermissionRequest>) | Api endpoint exposed to the frontend which calls the service to update the member permissions. |
| [ProjectService]  GetProjectMembers(int) | Get all members(registered, unregistered, teams) of a project. |
| [ProjectService] GetAllPermissions() | Get list of all project-type permissions. |
| [ProjectService]  GetAllUserPermissions(string) | Get all existing permissions of all users of a project. |
| [ProjectService] UpdateProjectPermissions(string, int?, List<UpdateProjectPermissionRequest>) | Update all members’ permissions of a project as modified by the owner. |

# Challenge Management

This feature includes challenge view including challenge list, adding/ removing in list etc.

## Challenges List

This includes retrieval all the challenges in the project with their data. The challenges which are not unlocked by the project owner will not be able to be accessed by anyone except the owner.

**Associated Database Tables**

* Project
* ProjectMember
* TemporaryProjectMember
* TeamMember
* TemporaryTeamProject
* Challenge
* ChallengeList
* UserChallengeDuration
* User

**Associated Files**

* ChallengeController.cs
* ChallengeService.cs
* ProjectDetailHeaderDto.cs
* ProjectMemberDto.cs
* ProjectDetailDto.cs
* ChallengeListDto.cs
* CardListDto.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [ChallengeController]  GetProjectDetailHeader(string) | Api endpoint exposed to the frontend which calls the service to get details for subheader of challenge screen. |
| [ChallengeController]  GetChallengeDetails(string) | Api endpoint exposed to the frontend which calls the service to get details of challenges of a project. |
| [ChallengeService]  GetProjectDetailHeader(string, int?) | Gets data of the project for the sub-header displayed on challenge(project detail) view. |
| [ChallengeService]  GetChallengeDetails(string, int?) | Gets all challenges of a project along with its lists, cards, etc. |

## Challenge-List Create/Delete

This feature includes the creation and deletion of challengelist inside a challenge. This feature permission based which is granted by the project owner. The permission is stored in “projectPermission” table in the db and the status of whether the permission is granted or not is stored in “projectMemberPermission” table.

**Associated Database Tables**

* ProjectMemberPermission
* ChallengeList

**Associated Files**

* ChallengeController.cs
* ChallengeService.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [ChallengeController]  AddListToChallenge(int?, string) | Api endpoint exposed to the frontend which calls the service to implement the business logic. |
| [ChallengeController]  DeleteList(int?) | Api endpoint exposed to the frontend which calls the service to implement the business logic. |
| [ChallengeService]  AddListToChallenge(int?, string, int?) | Add challenge-list with provided name to challenge specified by the challengeId. |
| [ChallengeService]  DeleteList(int? int?) | Soft-delete a challenge-list from challenge, listid provided as parameter in endpoint. |

## Card Create/Delete

This feature includes the creation and deletion of card inside a challengelist. This feature permission based which is granted by the project owner. The permission is stored in “projectPermission” table in the db and the status of whether the permission is granted or not is stored in “projectMemberPermission” table.  
An entry is also inserted in the activity log table for these actions.

**Associated Database Tables**

* ChallengeList
* Card
* ActivityLog

**Associated Files**

* ChallengeController.cs
* ChallengeList.cs
* AddActivityLogRequest.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [ChallengeController]  AddCardToList(int?, string) | Api endpoint exposed to the frontend which calls the service to add a card in a challengelist. |
| [CardController]  DeleteCard(int) | Api endpoint exposed to the frontend which calls the service to delete a card from a challengelist. |
| [ChallengeService]  AddCardToList(int?, string, int?) | Adds a new record in card table with the corresponding challengelist. |
| [CardService]  DeleteCard(int, int?) | Delete a card from the specified challengelist. |

## Actions

Card move from one list to another, unlocking a challenge and start/pause of user time on a challenge is included under this section.

Associated Database Tables

* ActivityLog
* Card
* ChallengeList
* Challenge
* UserChallengeDuration

Associated Files

* ChallengeController.cs
* ChallengeService.cs
* CardController.cs
* CardService.cs
* UserChallengeDuration.cs

Associated Methods

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [ChallengeController]  MarkChallengeComplete(int) | Api endpoint exposed to the frontend which calls the service to mark a challenge as complete. |
| [ChallengeController]  MarkChallengeIncomplete(int) | Api endpoint exposed to the frontend which calls the service to mark a challenge as incomplete. |
| [ChallengeController]  AddDuration(double, int) | Api endpoint exposed to the frontend which calls the service to add work duratio ofr a user against a challenge. |
| [CardController]  MoveCardInList(int, int) | Api endpoint exposed to the frontend which calls the service to update the challengelist for a card moved between lists. |
| [ChallengeService]  MarkChallengeComplete(int, int?) | Marks the specified challenge as completed and unlocks the next challenge if exist. |
| [ChallengeService]  MarkChallengeIncomplete(int, int?) | Marks the specified challenge as incomplete. |
| [ChallengeService]  AddDuration(double, int, int?) | Adds the duration of work for the user against a challenge to existing record or create a new record for it if not exist. |
| [CardService]  MoveCardInList(int, int ,int?) | Updates the challengelist reference for the card moved. |

# Card Management

Card management includes all th functionalities of the card view e.g. asisgn member, attachment, checklist, etc.

## GetDataOnLoad

This includes the endpoint(s) that are used to get data of the card.

**Associated Database Tables**

* Card
* CardAssignedMember
* CardAttachment
* Comment
* ActivityLog
* Checklist

**Associated Files**

* ProjectController.cs
* CardController.cs
* ChallengeController.cs
* ProjectService.cs
* CardService.cs
* ChallengeService.cs
* ProjectPermissionAllowedDto.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [ProjectController]  GetProjctMemberPermissions(string) | Api endpoint exposed to the frontend which calls the service to get the project permissions for the member. |
| [CardController]  GetCardDetails(int?) | Api endpoint exposed to the frontend which calls the service to get all of the card’s data. |
| [ChallengeController]  GetProjectDetailHeader(string) | Api endpoint exposed to the frontend which calls the service for members. |
| [ProjectService]  GetProjectMemberPermissions(string, int?) | Gets the permissions of the members of the specified project. |
| [CardService]  GetCardDetails(int?) | Get card’s data like attachment, checklist, etc. |

## Assign Member

This feature allows the user to assign a member to a card. This feature permission based which is granted by the project owner. The permission is stored in “projectPermission” table in the db and the status of whether the permission is granted or not is stored in “projectMemberPermission” table.

**Associated Database Tables**

* Card
* CardAssignedMember
* User

**Associated Files**

* CardController.cs
* CardService.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [CardController]  AssignMember(string, int) | Api endpoint exposed to the frontend which calls the service to add a member in a card. |
| [CardController]  UnassignMember(string, int) | Api endpoint exposed to the frontend which calls the service to remove a member from the card. |
| [CardService]  AssignMember(string, int, int?) | Adds a member specified by its email in the card speicifed by its id. |
| [CardService]  UnassignMember(string, int, int?) | Removes a member specified by its email from the card specified by its id. |

## Set Due-Date

This feature allows the user to set due-date for a card. This feature permission based which is granted by the project owner. The permission is stored in “projectPermission” table in the db and the status of whether the permission is granted or not is stored in “projectMemberPermission” table.

**Associated Database Tables**

* Card

**Associated Files**

* CardController.cs
* CardService.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [CardController]  SetDueDate(int, string) | Api endpoint exposed to the frontend which calls the service to set the due-date for the card. |
| [CardService]  SetDueDate(int, string, int?) | Sets the due-date for the card specified by its id. |

## Change Card Status

This feature allows the user to set the status of the card as either ‘complete’ or ‘inprogress’.

**Associated Database Tables**

* Card

**Associated Files**

* CardController.cs
* CardService.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explalation** |
| [CardController]  ChangeCardStatus(int, int) | Api endpoint exposed to the frontend which calls the service to set the status of the card. |
| [CardService]  ChangeCardStatus(int, int, int?) | Changes the status of the specified card with the value specified by its enum integer. |

## Checklist/ Sub-Checklist

Allows the user add, remove or modify the checklist, add sub-checklist to a list, modify or remove that sub-checklist.

**Associated Database Tables**

* CheckList

**Associated Files**

* CardController.cs
* CardService.cs
* AddCheckListRequest.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [CardController]  AddCheckList(AddCheckListRequest) | Api endpoint exposed to the frontend which calls the service add a checlist in a card. |
| [CardController]  DeleteCheckGroup(int) | Api endpoint exposed to the frontend which calls the service to delete a checklist group with a ll of its checklists. |
| [CardController]  DeleteCheckItem(int) | Api endpoint exposed to the frontend which calls the service to to delete a checklist with a ll of its sub-checklists. |
| [CardController]  DeleteCheckSubItem(int) | Api endpoint exposed to the frontend which calls the service to to delete a sub-checklist item. |
| [CardService]  AddCheckList(AddCheckListRequest, int?) | Adds an checklist item in a card. |
| [CardService]  DeleteCheckGroup(int, int?) | Removes all the checklist under a checklist group. |
| [CardService]  DeleteCheckItem(int, int?) | Removes a checklist and all of its subitems. |
| [CardService]  DeleteCheckSubItem(int, int?) | Removes a sub-checklist item. |

## Attachment

This feature allows the user to add or remove an attachment from a card. This feature is permission based which is granted by the project owner. The permissions are stored in “projectPermission” table in the db and the status of whether those permissions are granted or not is stored in “projectMemberPermission” table.

**Associated Database Tables**

* CardAttachment

**Associated Files**

* CardController.cs
* CardService.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [CardController]  AddAttachment(int, IFormFile) | Api endpoint exposed to the frontend which calls the service to add an attachment in a card. |
| [CardService]  AddAttachment(int, IFormFile, int?) | Adds an attachment for a card and uploads the file on the server. |

## Comment

Allows the user to add or remove a comment in a card.

**Associated Database Tables**

* Comment

**Associated Files**

* CardController.cs
* CardService.cs
* AddCommentRequest.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [CardController]  AddComment(AddCommentRequest) | Api endpoint exposed to the frontend which calls the service to add a record for a comment. |
| [CardController]  DeleteComment(int?) | Api endpoint exposed to the frontend which calls the service to remove a record for a comment. |
| [CardService]  AddComment(AddCommentRequest, int?) | Adds a comment in a card against a user. |
| [CardService]  DeleteComment(int?, int?) | Removes a comment from a card of a user. |

## Activity Log

Keeps track of the activities in a card i.e. card move between list, attachment add or remove, checklist add,remove or modification, etc.

**Associated Database Tables**

* Activity Log

**Associated Files**

* CommonController.cs
* CommonService.cs
* AddActivityLogRequest.cs

**Associated Methods**

|  |  |
| --- | --- |
| **Method** | **Explanation** |
| [CommonController]  AddActivityLog(AddActivityLogRequest) | Api endpoint exposed to the frontend which calls the service to add a new record for the activity. |
| [CommonService]  AddActivityLog(AddActivityLogRequest, int?) | Inserts a new record for the activity according to the data object. |