Create an asp.net mvc app, called "Reporting Assistant", for "Task Assignment and tracking the progress of task".

The company has two types of users:  "User" and "Admin".

The admin user assigns one or more tasks to the user everyday.

The users can see the tasks assigned to them [from different admins] and start doing those tasks. [It will be stored in "Tasks" table]

At the end of day, the user can add the information about the tasks he has done during the day. [It will be stored in "TasksDone" table]

At the end of same day or next day morning, the admin user will re-check the tasks completed by the user and add his comments about the pending work, bugs, issues, approvals or any other comments. [It will be stored in "FinalComments" table]

The user will check the "FinalComments" in the next day and do work through the day and add the details of tasks completed to "TasksDone" table.

Any admin can add tasks to any user.

So the user may get tasks from different admins; multiple tasks from the same admin also.

All tasks done by all users, are visible to all admins.

Any admin can add comments etc., to any task done by any user [will be stored in "finalcomments" table].

The user can edit "tasks done" details of the same user.

The admin can edit "tasks" and "final comments" of the same admin; but can view all "tasks", "tasks done" and "final comments" of all users and all admins.

No one can assign tasks to "admin" users.

One person can play role of "user" and "admin", at-a-time; then other admins can assigns tasks to him, as he plays a role of "User".

Each "task", "task done" and "final comment", have an option of attachment [image].

**Roles:**

1. "User" role

      By default, it must have three users: "nuzhath", "harsha", "nazareen".

2. "Admin" role  
      By default, it must have three admins:  "harsha", "shylender", "harika".

Note:  "harsha" is single person; but plays the role of "Admin" and "User".

**Tables:**

**1. "Categories" table:**

   CategoryID: long, primary key, not null, unique, auto-generated.

   CategoryName: string, allow alphabets, digits and spaces only, not null, unique.

**2. "Projects" table:**

   ProjectID: long, primary key, not null, unique, auto-generated

   ProjectName: string, allow alphabets, digits and spaces only, not null, unique

   DateOfStart: datetime, allow null

   AvailabilityStatus: Only two options: "Available" and "Not Available"; radio button

   CategoryID: Foreign key from "Categories" table, shown as dropdownlist; not null

   Photo: Allow file uploading; store image in some folder and store the image path in this column; allow null.

**3. "Tasks" table:**

   TaskID: long, primary key, not null, unique, auto-generated

   Screen: string, min 2 characters length, max 50 characters.

   Description: string, min 2 characters length, max 10000 characters.

   AdminUserID: long, Foreign key from "AspNetUsers" table of Asp.Net Identity; not null

   UserID: long, Foreign Key from "AspNetUsers" table of Asp.Net Identity; not null

   DateOfTask: datetime, not null

   Attachment:  Allow file uploading; store image in some folder and store the image path in this column; allow null.

   ProjectID: Foreign Key from "Projects" table

**4. "TasksDone" table:**

   TaskDoneID: long, primary key, not null, unique, auto-generated

   Screen: string, min 2 characters length, max 50 characters.

   Description: string, min 2 characters length, max 10000 characters.

   UserID: long, Foreign Key from "AspNetUsers" table of Asp.Net Identity; not null

   DateOfTaskDone: datetime, not null

   Attachment:  Allow file uploading; store image in some folder and store the image path in this column; allow null.

   ProjectID: Foreign Key from "Projects" table

**5. "FinalComments" table:**

   FinalCommentID: long, primary key, not null, unique, auto-generated

   Screen: string, min 2 characters length, max 50 characters.

   Description: string, min 2 characters length, max 10000 characters.

   AdminUserID: long, Foreign key from "AspNetUsers" table of Asp.Net Identity; not null

   UserID: long, Foreign Key from "AspNetUsers" table of Asp.Net Identity; not null

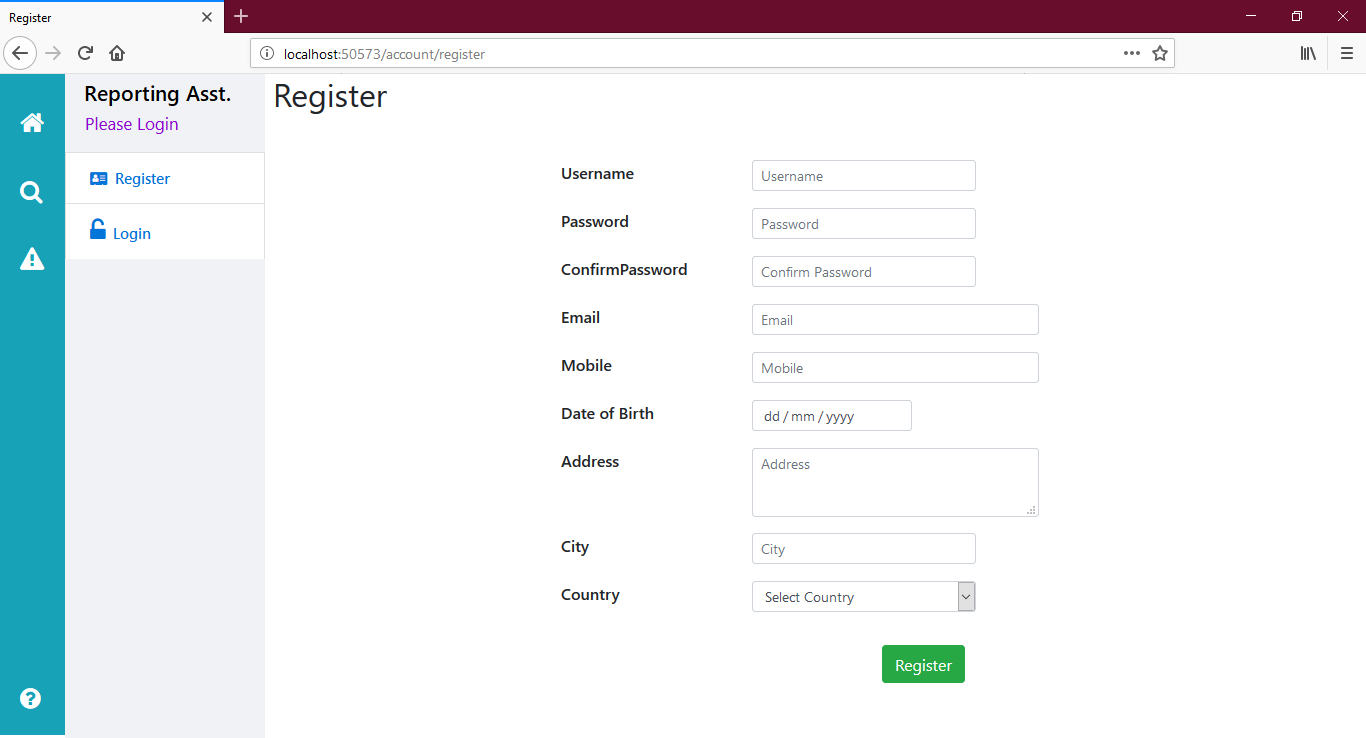
   DateOfFinalComment: datetime, not null

   Attachment:  Allow file uploading; store image in some folder and store the image path in this column; allow null.

   ProjectID: Foreign Key from "Projects" table

**Screens:**

**Screen 1: Registration Page**

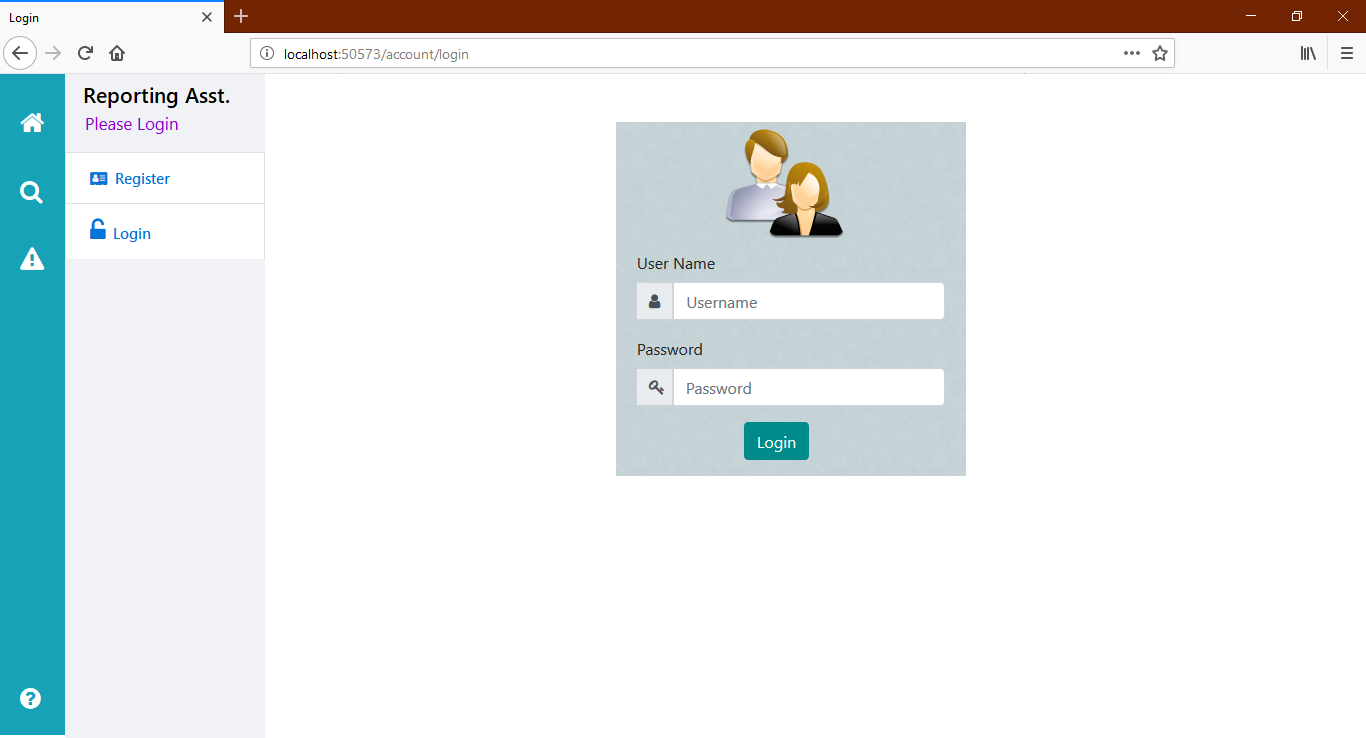


Using this screen, the user can register.

The "username", "password", "confirm password", "email" are mandatory fields.

Also make general validations like checking the pattern of password, email, mobile and date of birth.

**Screen 2: Login Page**

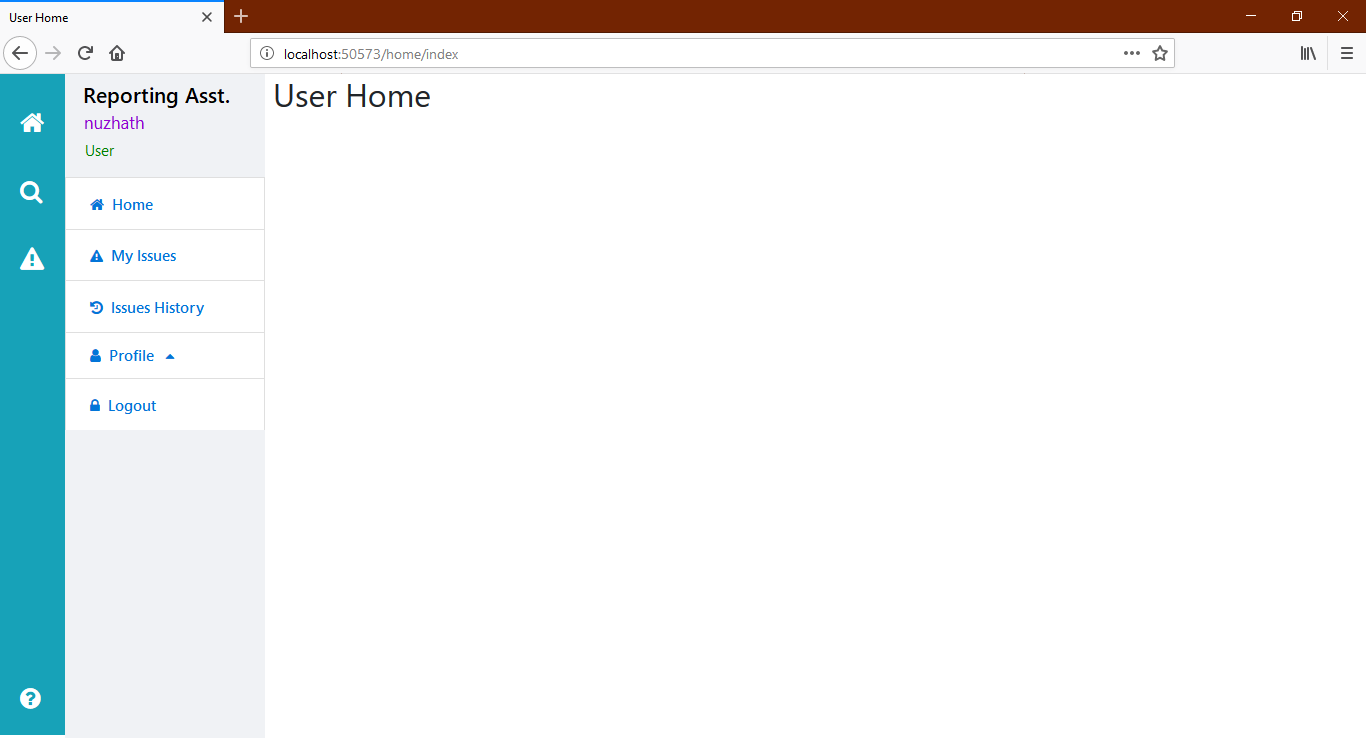


It is a common login page for both admin's and user's.

If he is an admin, redirect to "admin home page" of "admin area".

If he is a "user", redirect to "user home page".

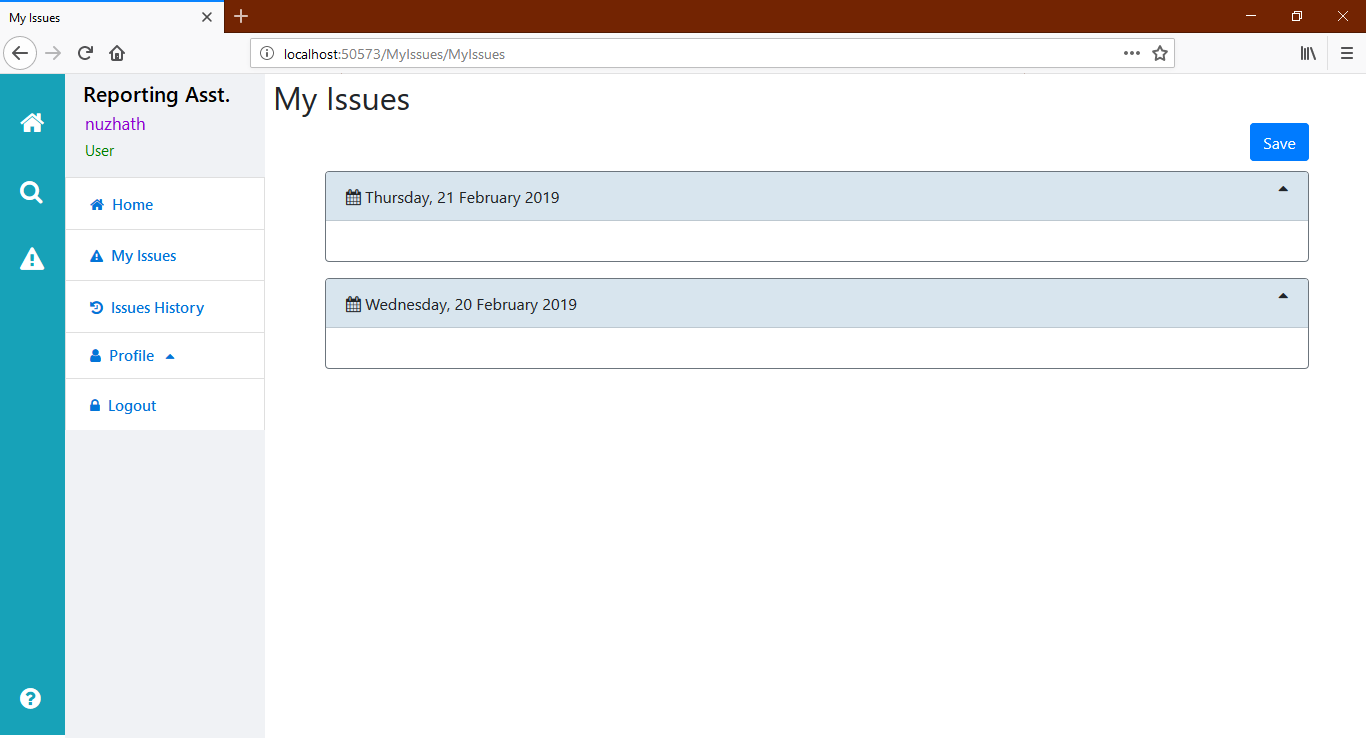
**Screen 3: User - Home Page**



After successful login, this user home page appears.

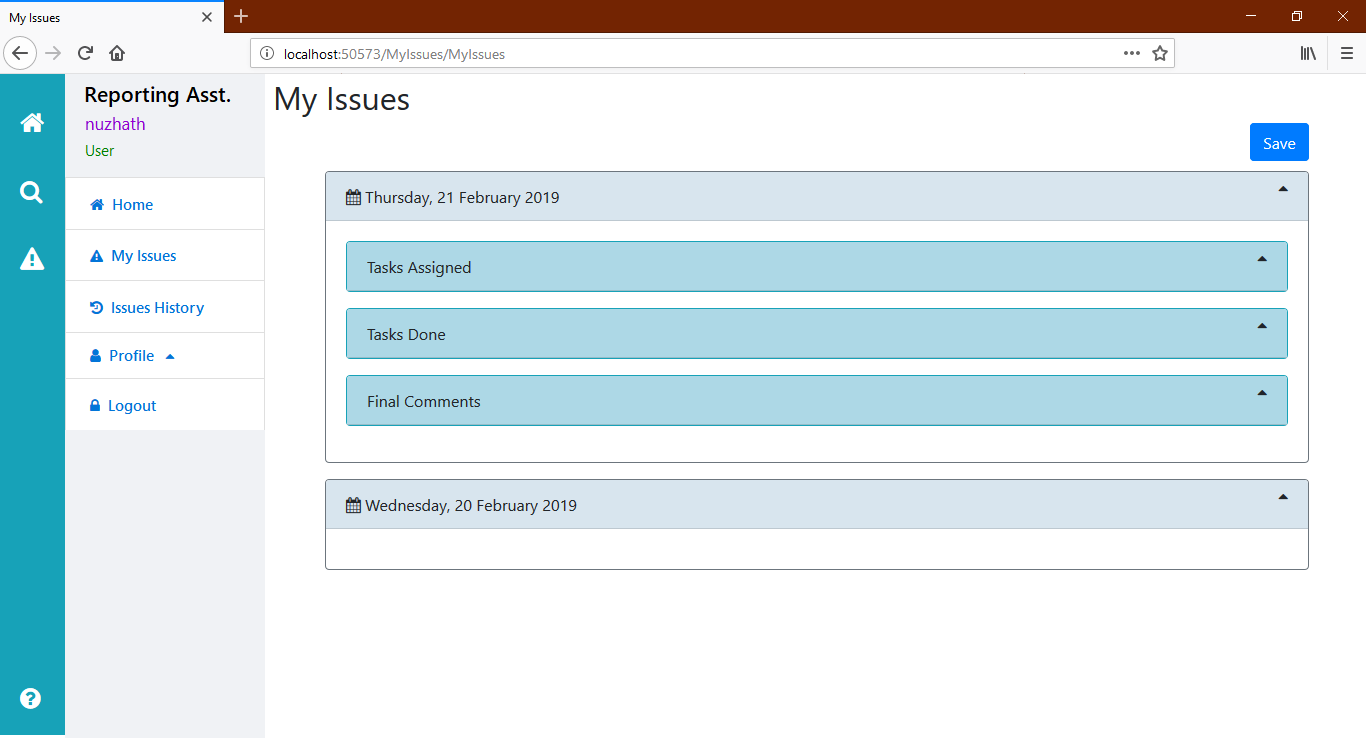
The current user name should appear as shown in the image.

**Screen 4: User - My Issues Page**



Once this "My Issues" page is opened, it has to display the two dates (today and yesterday), automatically.

**Screen 5: User - My Issues Page - Date**



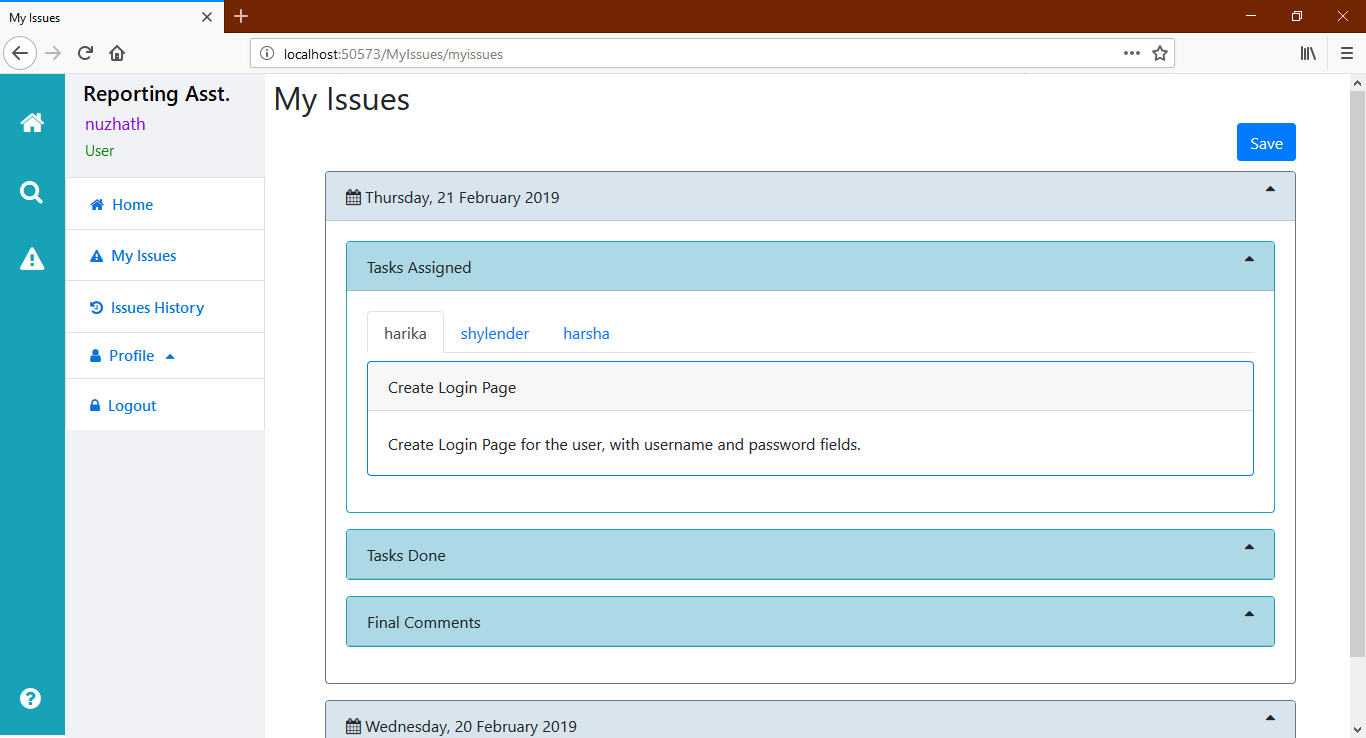
Once he clicks on a date (collapsible in bootstrap), it has to display three options:

"Tasks Assigned", "Tasks Done" and "Final Comments".

Even these three options also collapsible items.

[Use cards and collapsible components in bootstrap].

**Screen 6: User - My Issues Page - Tasks Assigned**

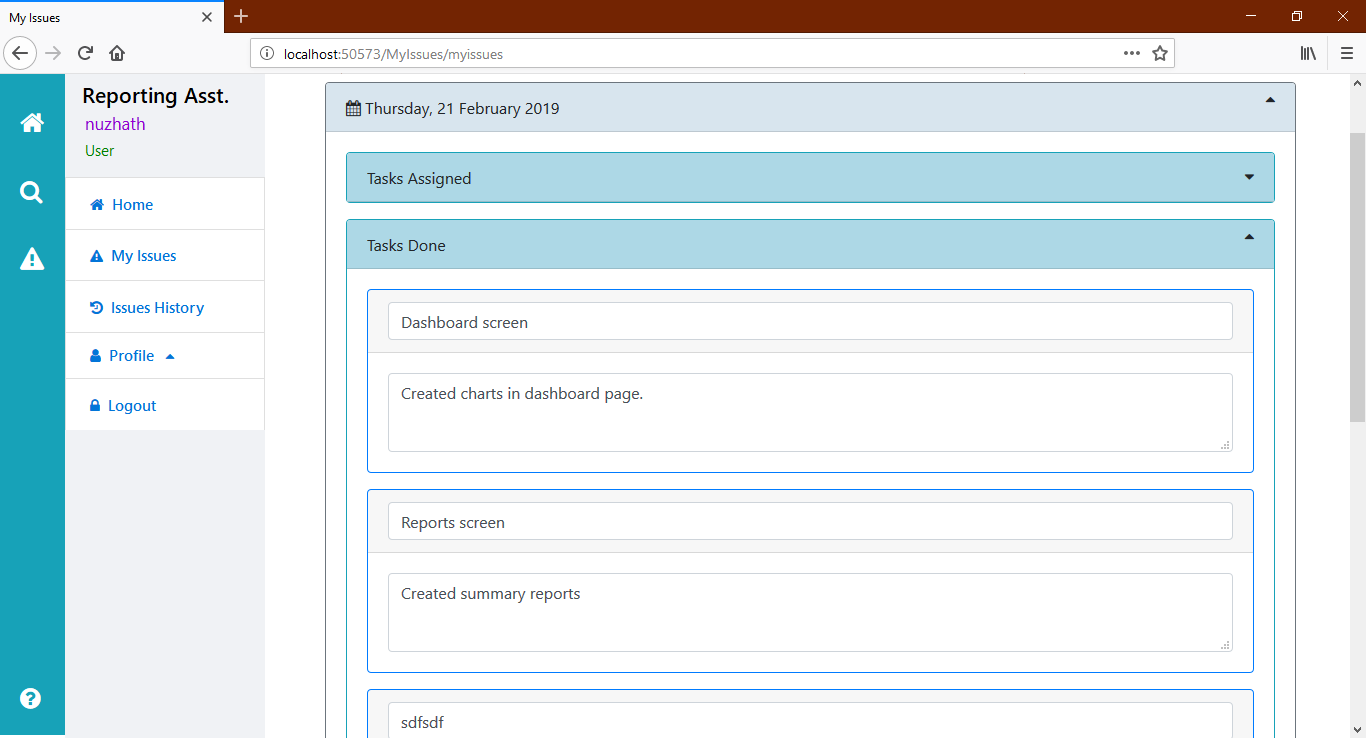


The admin names [who have given tasks to the current user] are shown as tabs

In each tab of admin, all the tasks assigned by that admin are shown.

For example, in "harika" (is admin), the list of tasks assigned by "harika" to "nuzhath" are shown.

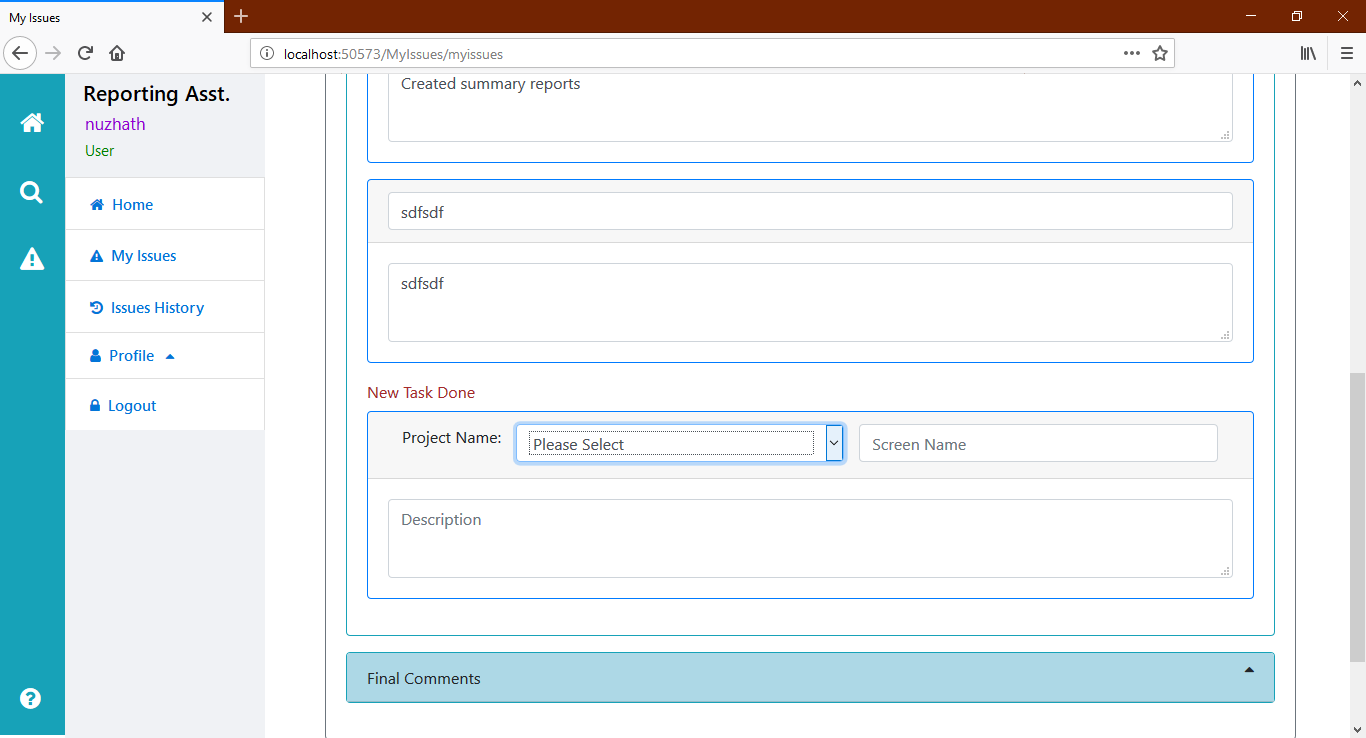
**Screen 7: User - My Issues Page - Tasks Done**



Under the "Tasks Done" collapsible card, it has to display the list of existing "tasks done" records, given by the current user.

It shows screen name, and description. It would be better if project name also displayed.

**Screen 8: User - My Issues Page - New Task Done**

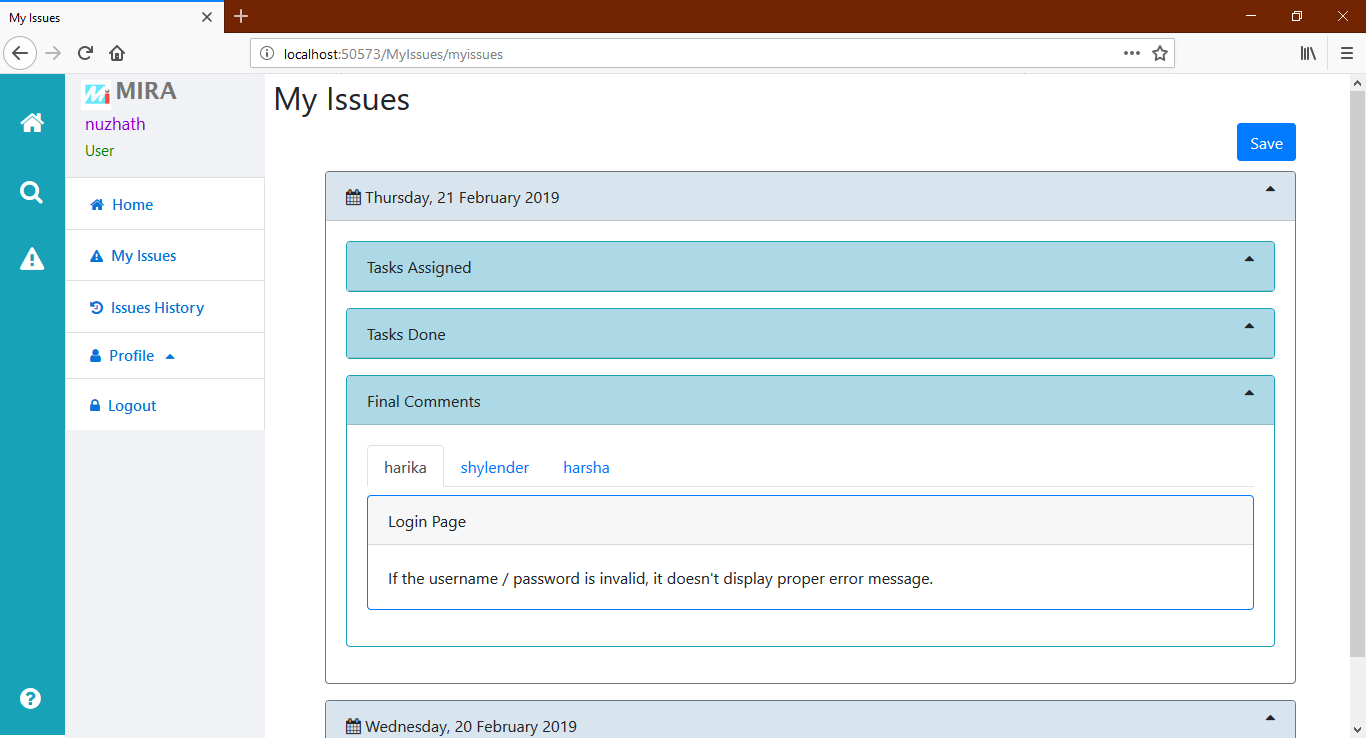


The user can select the project name from dropdown (loaded from projects table), enter the screen name and description and click on "Save" at above.

When the user clicks on "Save", the existing records of "Tasks Done" should be updated; and also new "task done" should be inserted, if the description is not empty.

It is not must to add new task, at each time of clicking "Save" button. One can click on "Save" only to edit existing records of "tasks done".

**Screen 9: User - My Issues Page - Final Comments**

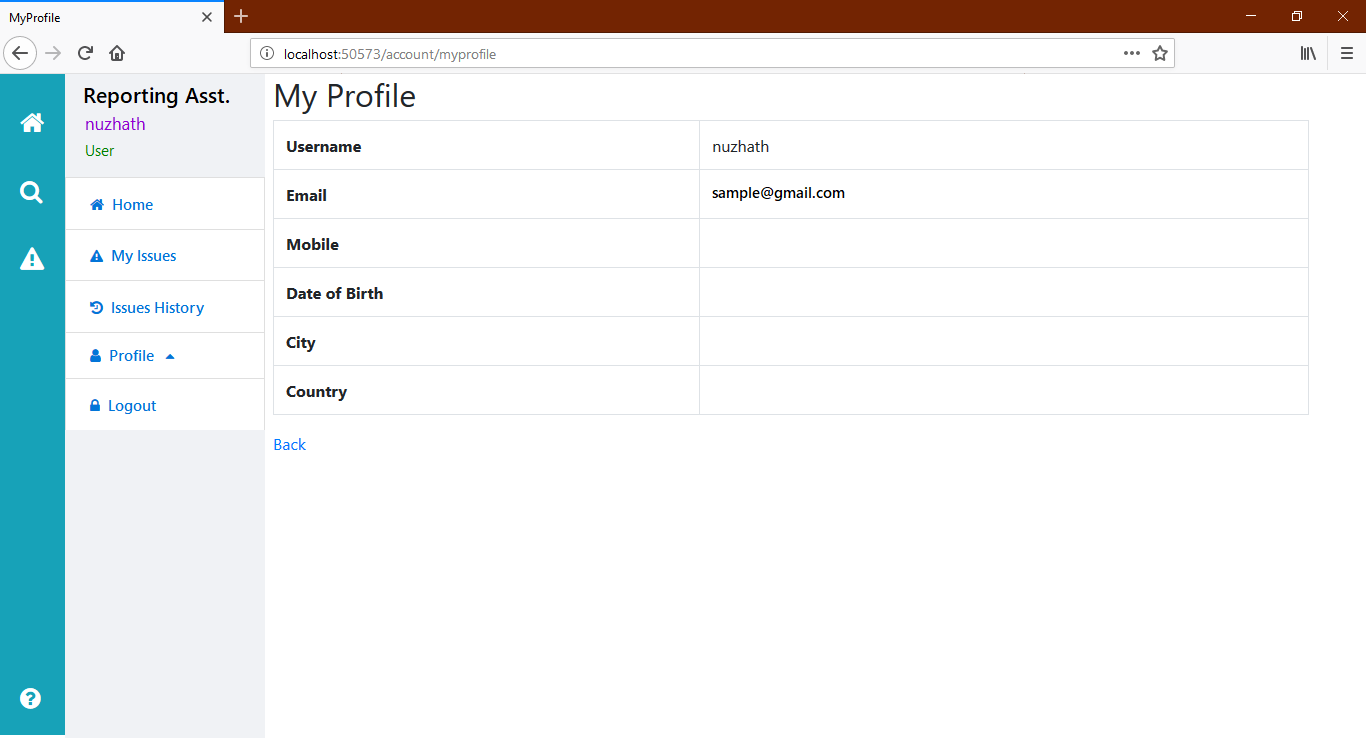


Display the existing records of "final comments", group-up by admin, in the form of bootstrap tabs, just like "Tasks assigned".

Same kind of output for other dates also [Today + Yesterday].

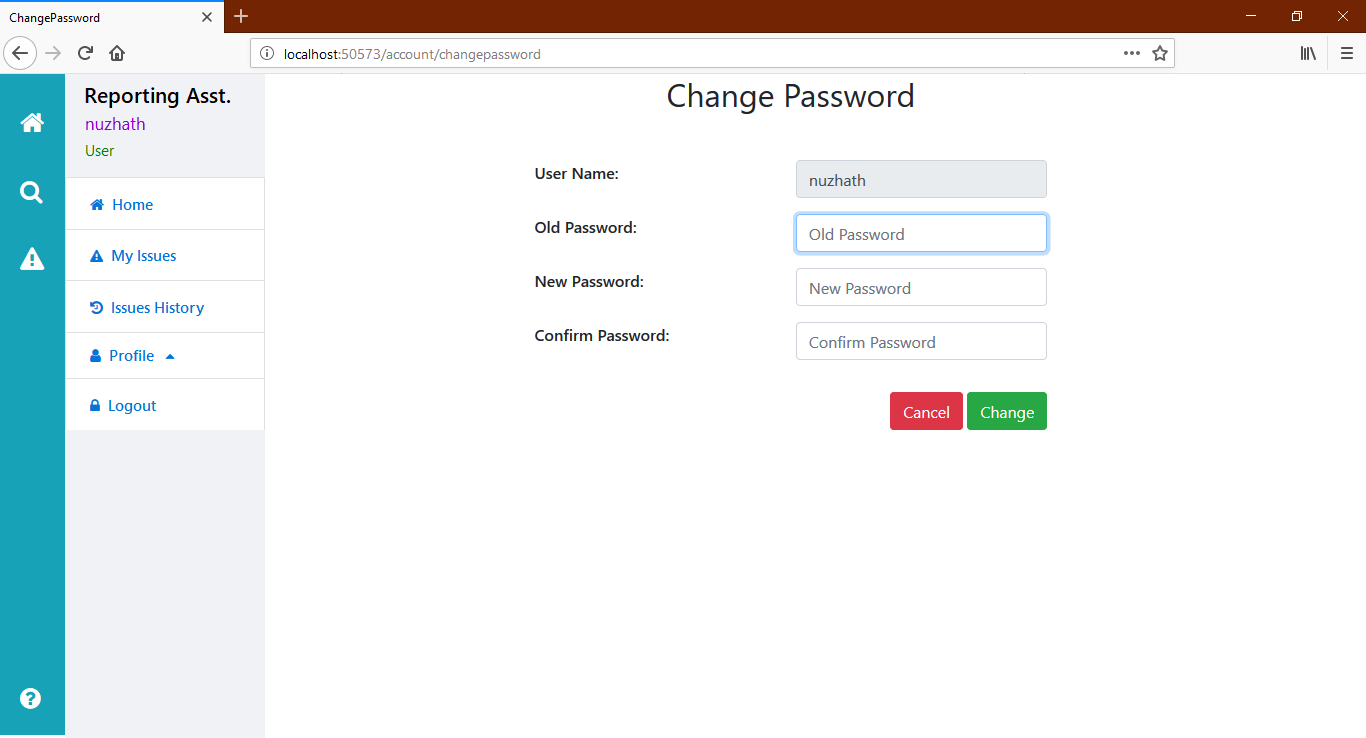
The user can modify, add "tasks done" details of both dates at-a-time and click on "Save".

**Screen 10: User - My Profile Page**



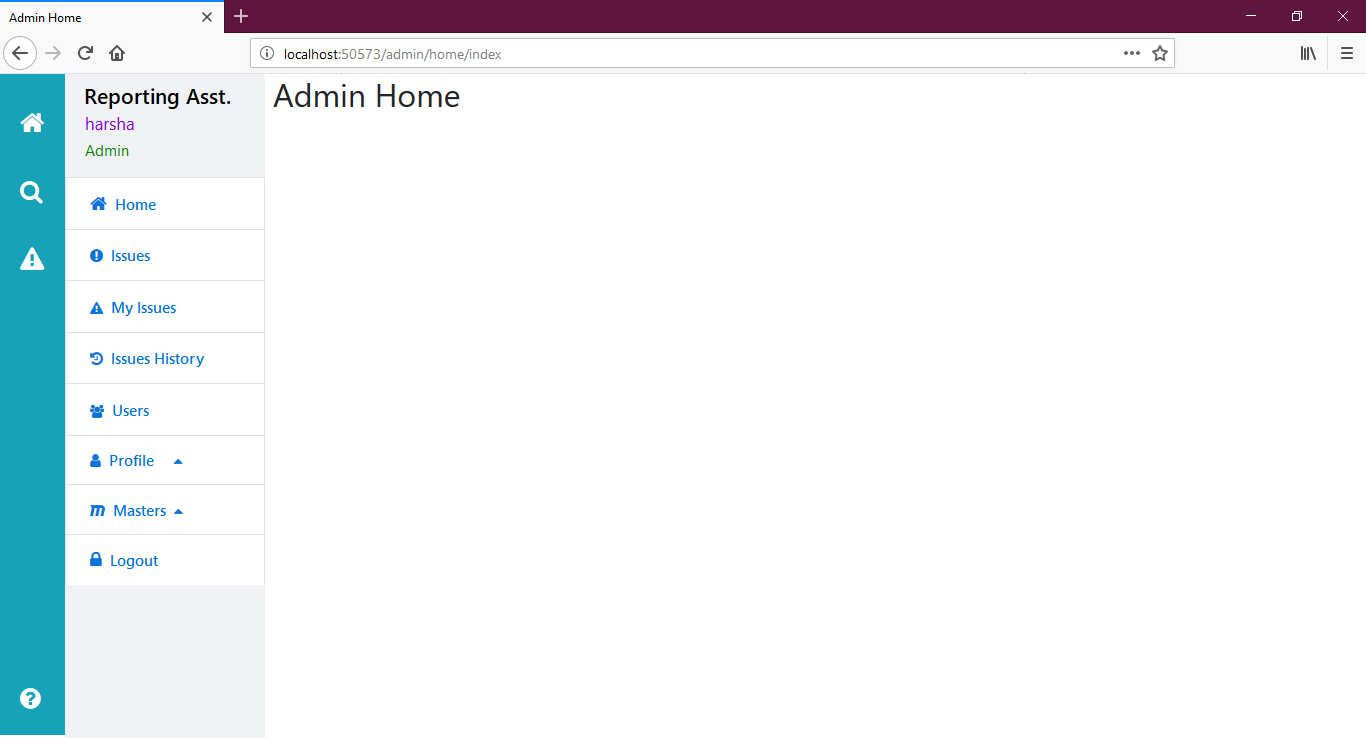
Display the profile of current user.

**Screen 11: User - Change Password**

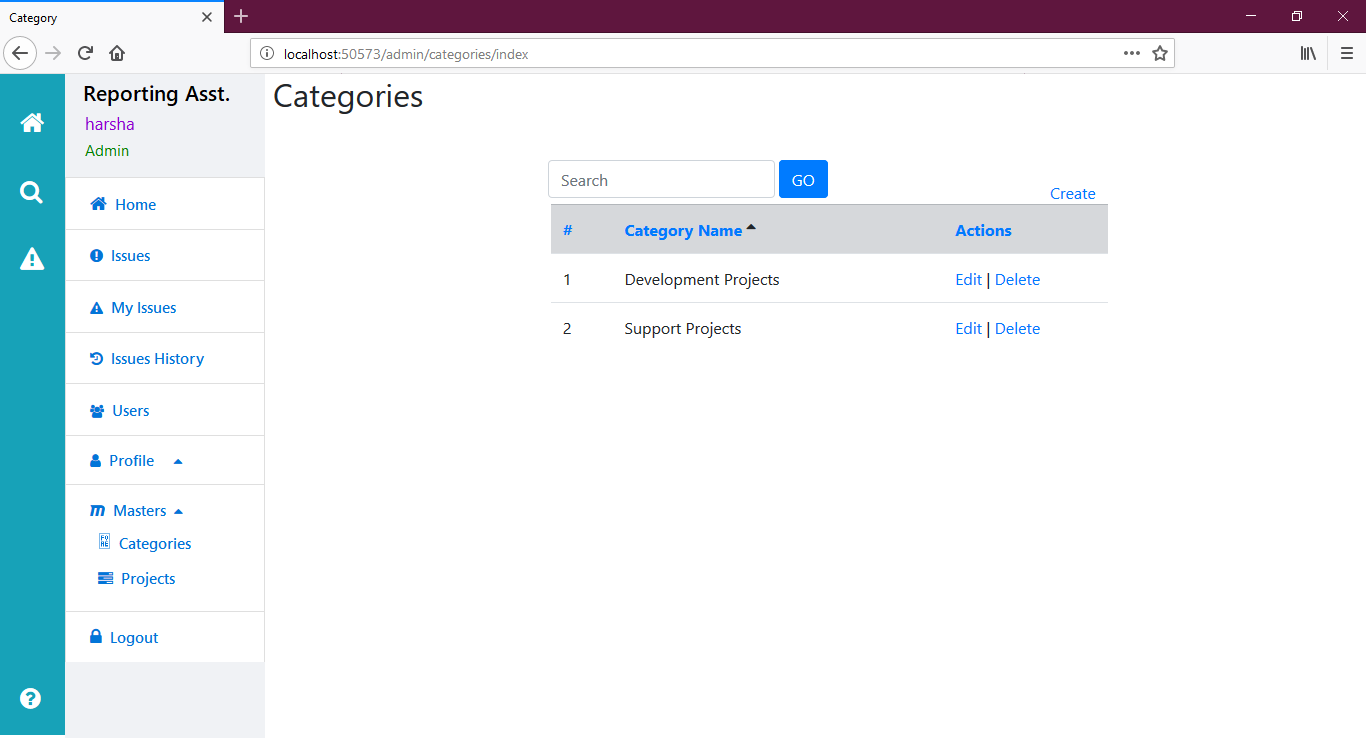


**Screen 12: Admin - Home Page**

In case of admin, it should redirect from "Login" page to the "admin home page".

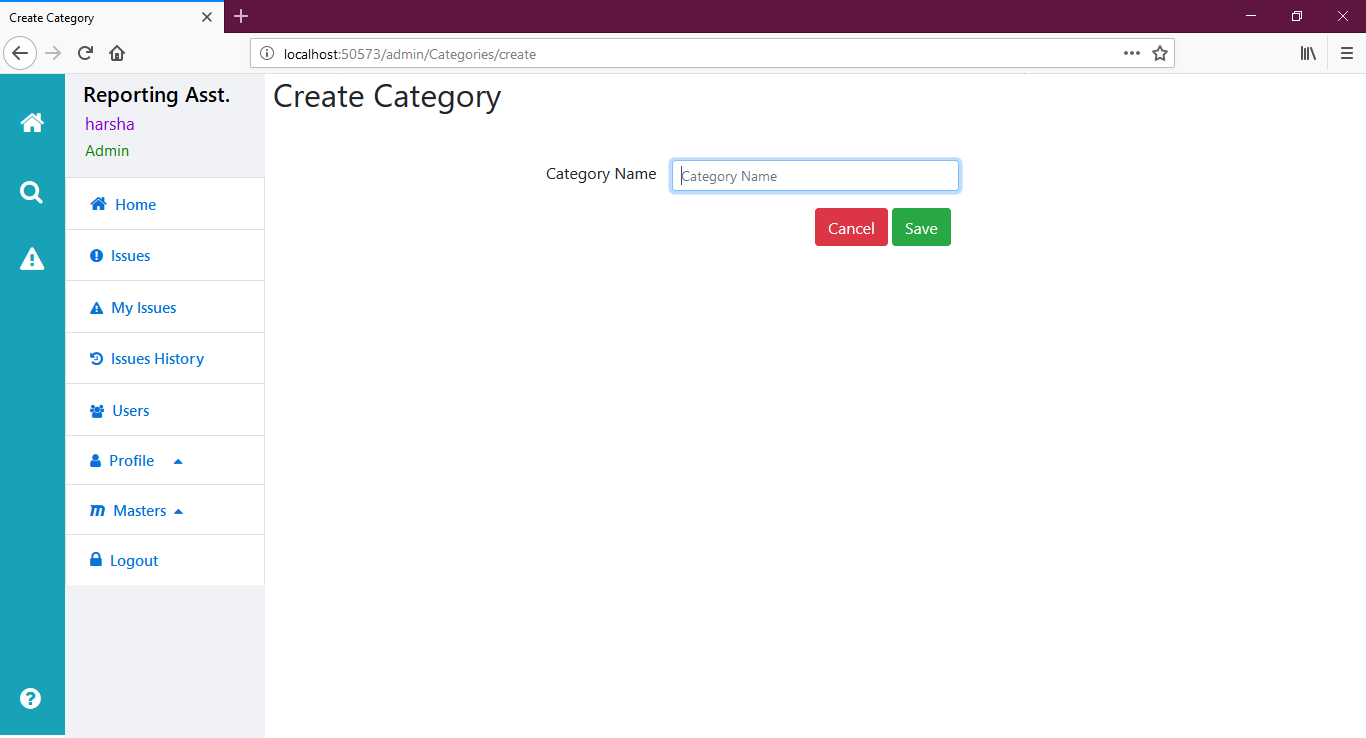


**Screen 13: Admin - Categories Page**



Display existing categories.

**Screen 14: Admin - Crate Category page**

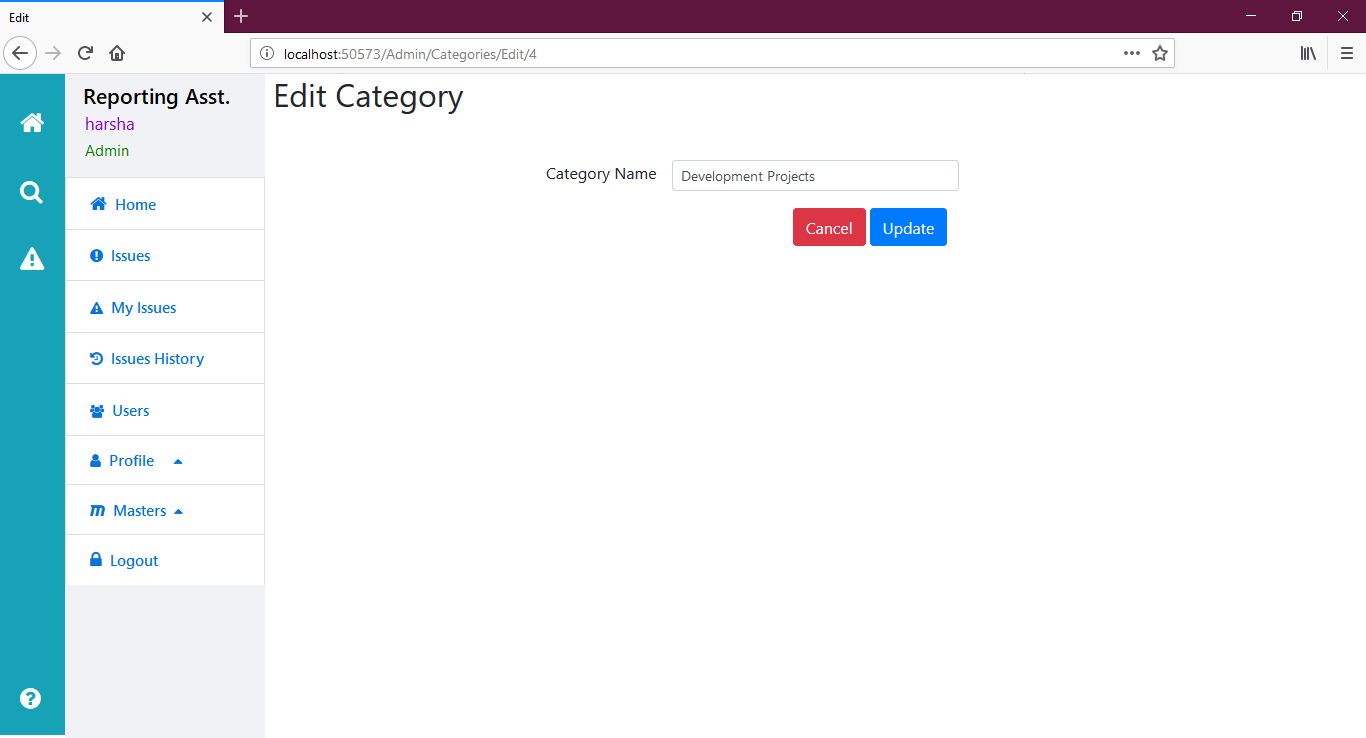


Admin can add category;

Redirect to "Categories" page, after inserting.

Add necessary validations, as explained above.

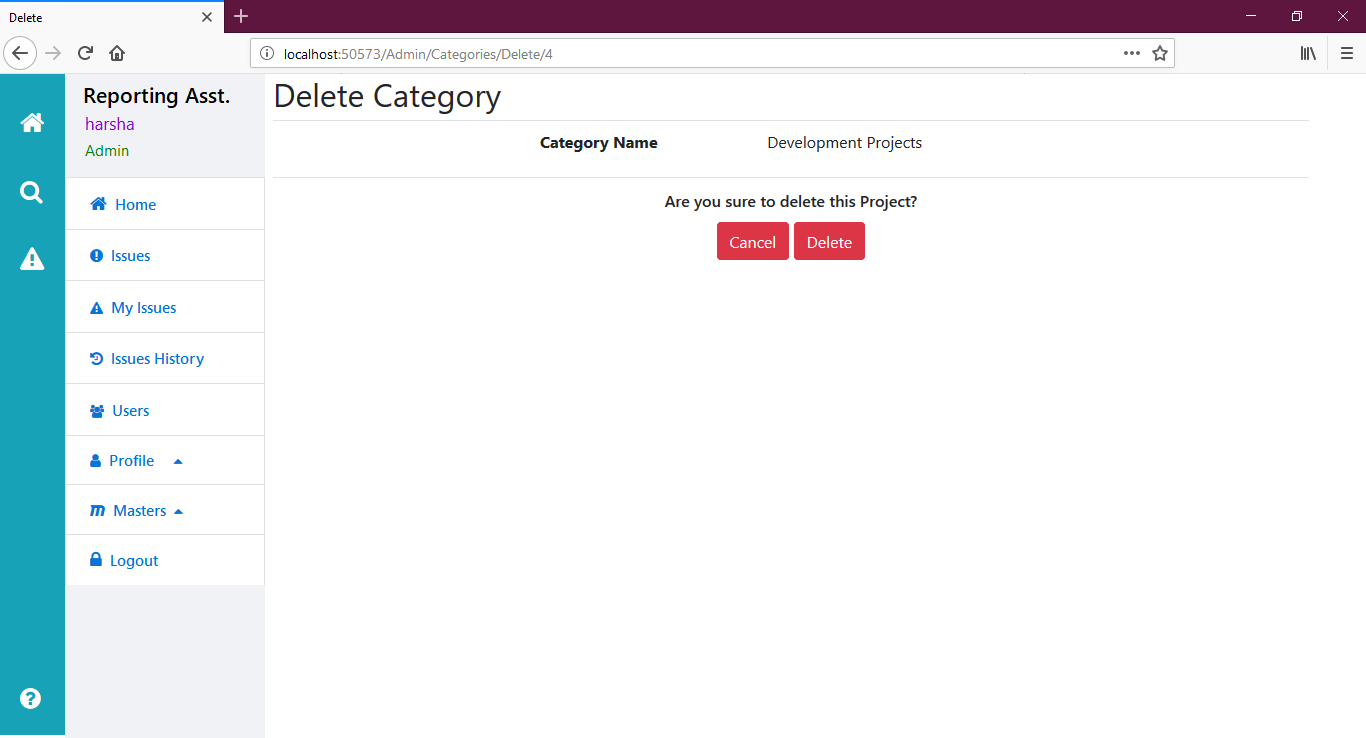
**Screen 15: Admin - Edit Category Page**



Category can be edited.

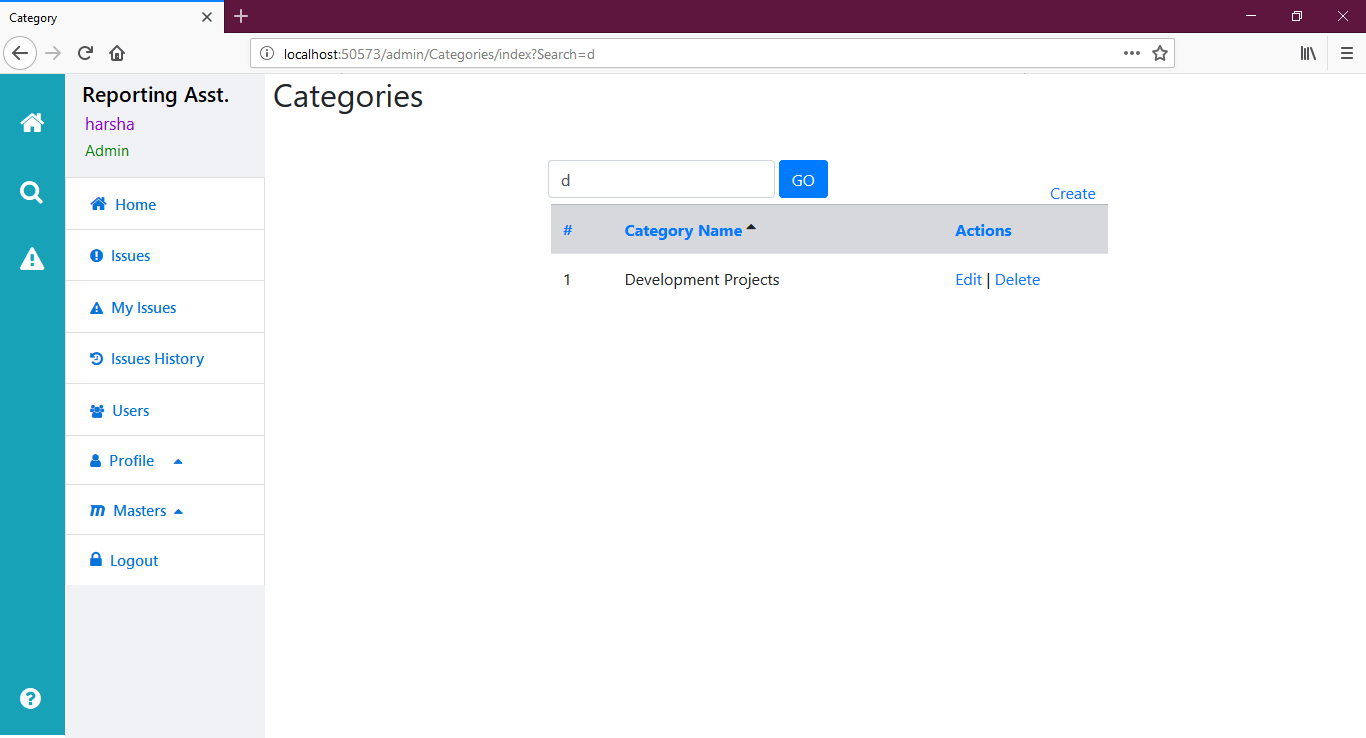
Add necessary validations, as explained above.

**Screen 16: Admin - Delete Category Page**



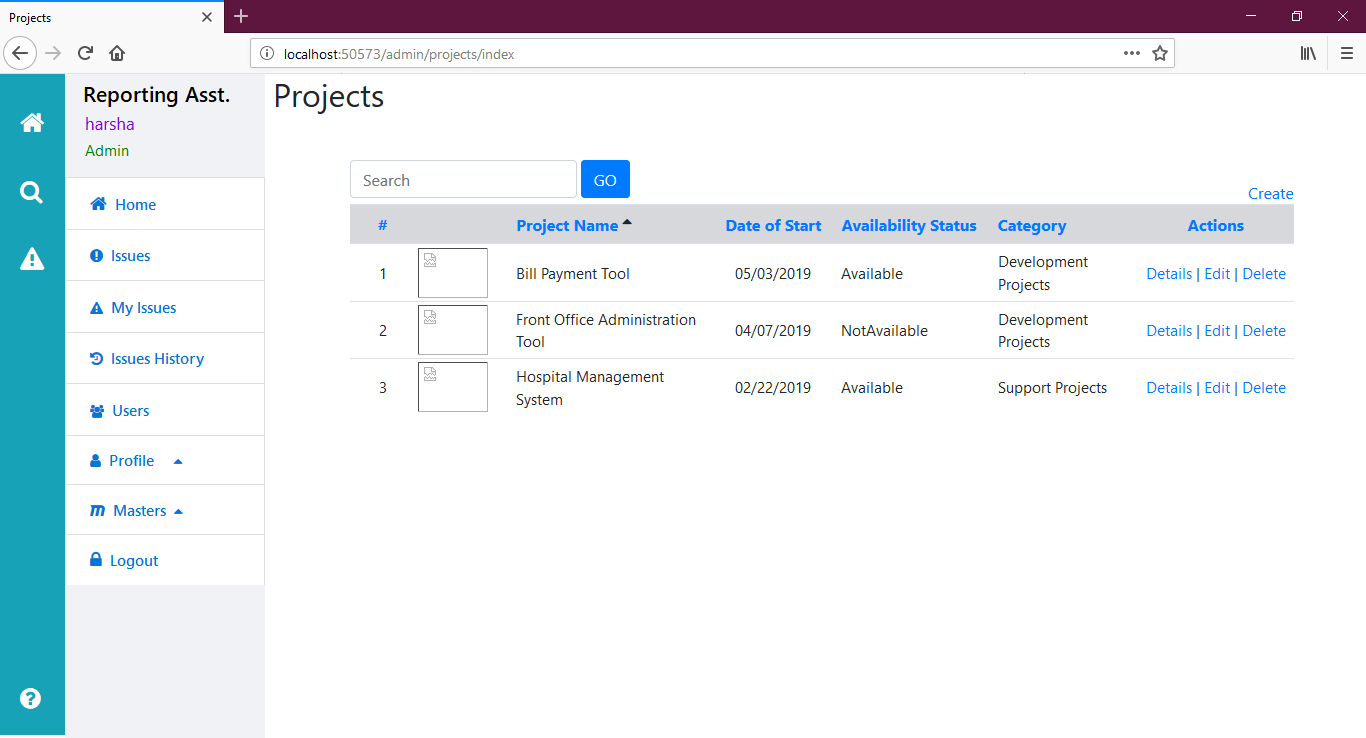
Admin can delete the category, only if it has no projects.

**Screen 17: Admin - Search Categories**



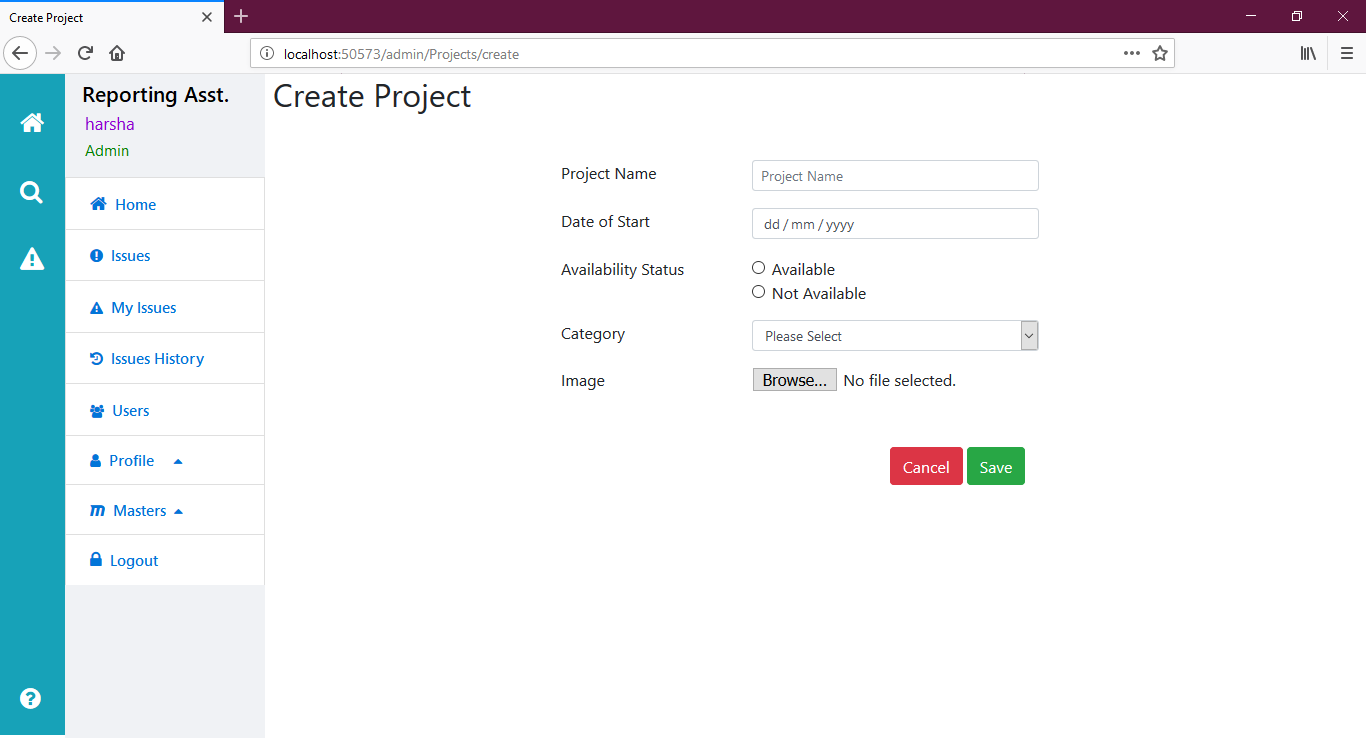
Categories are to be displayed based on the search characters.

**Screen 18: Admin -  Projects Page**



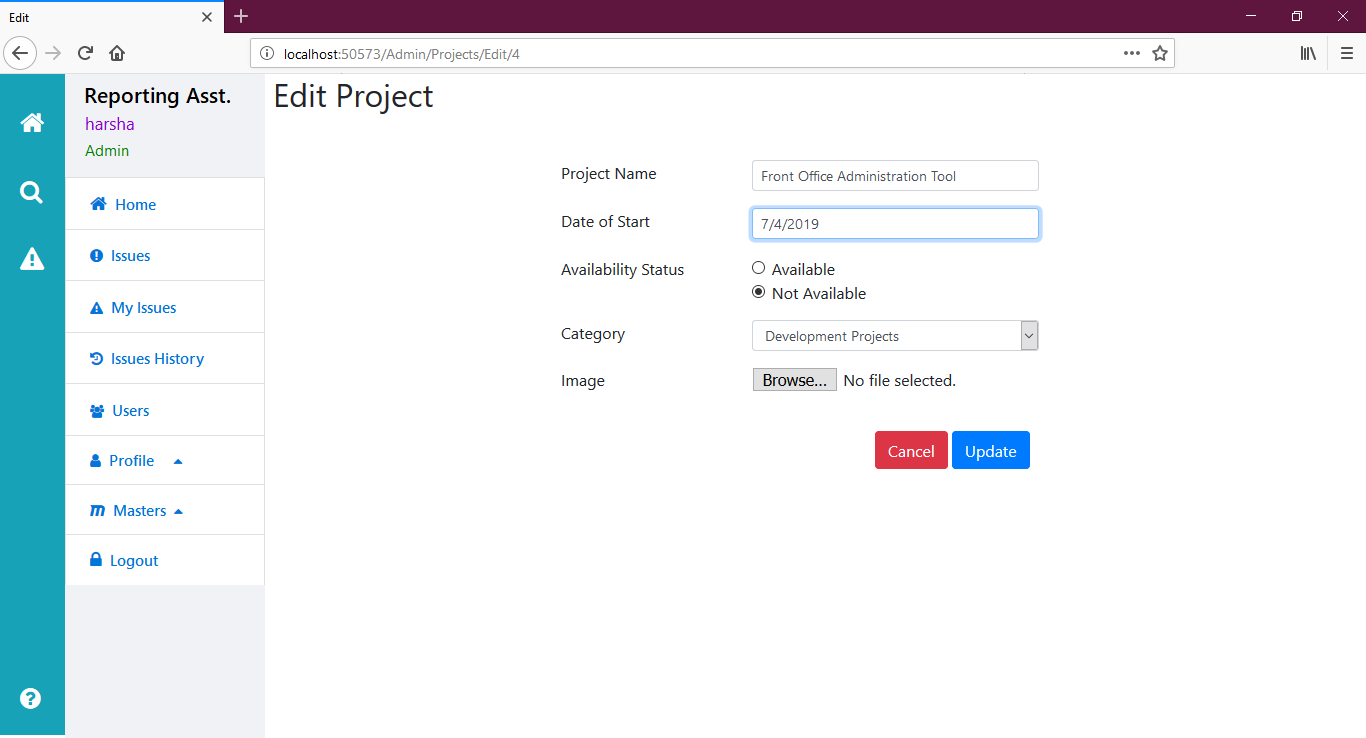
Display existing projects in the grid.

**Screen 19: Admin - Create Project Page**



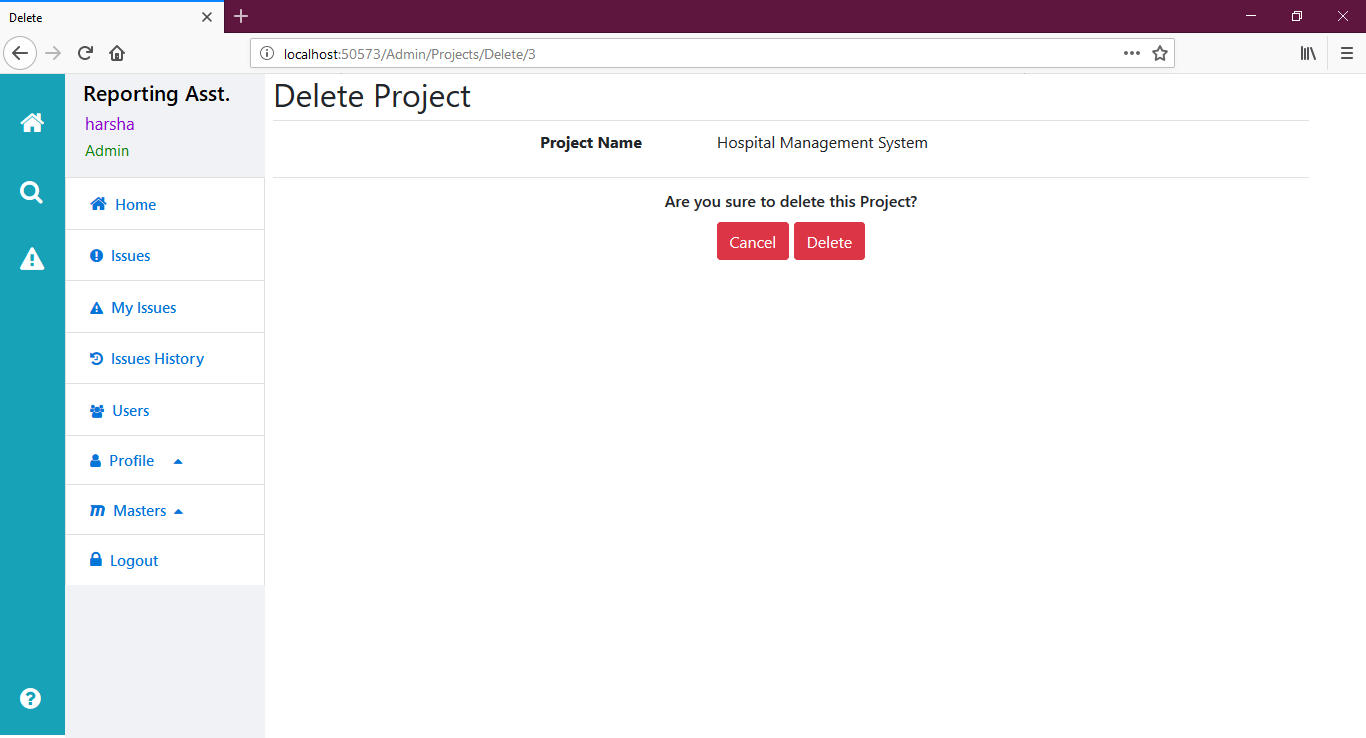
Add necessary validations, as explained above, with image uploading.

**Screen 20: Admin - Edit Project Page**



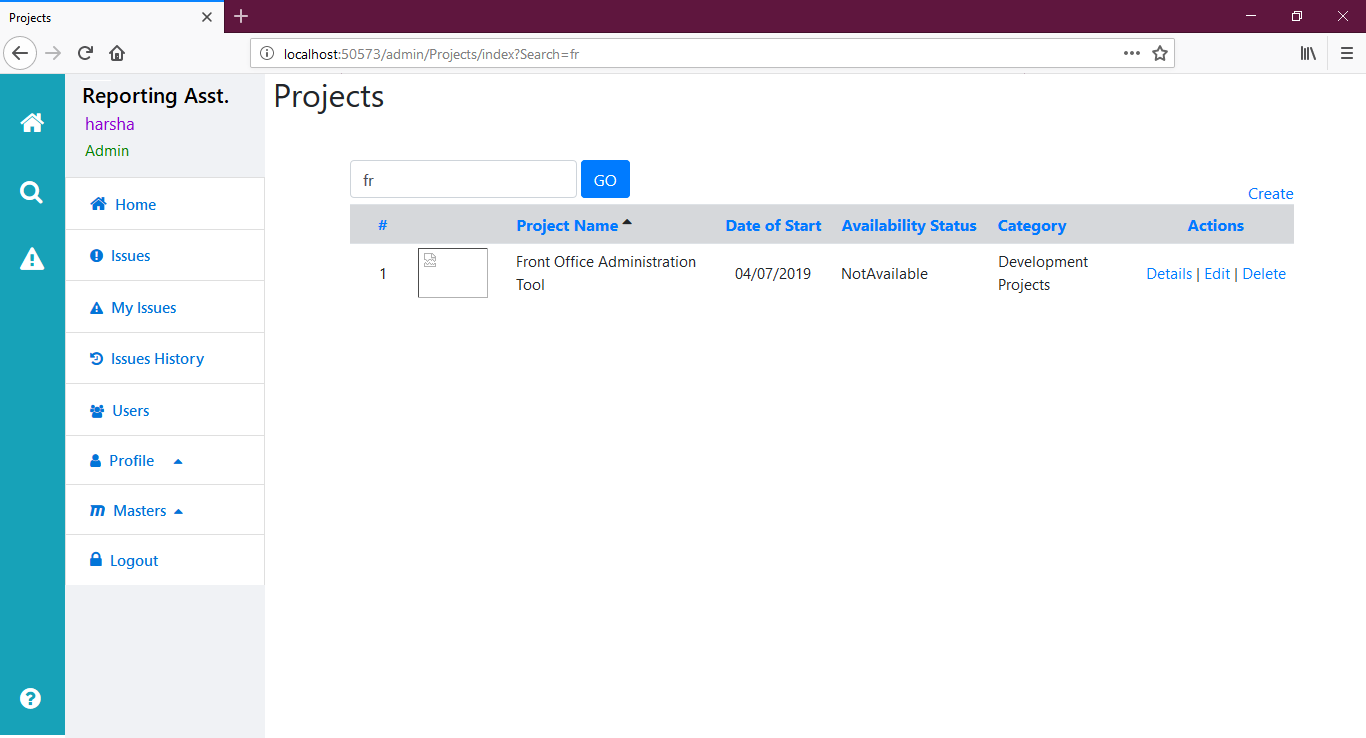
Create all validations, with file uploading.

**Screen 21: Delete Project Page**



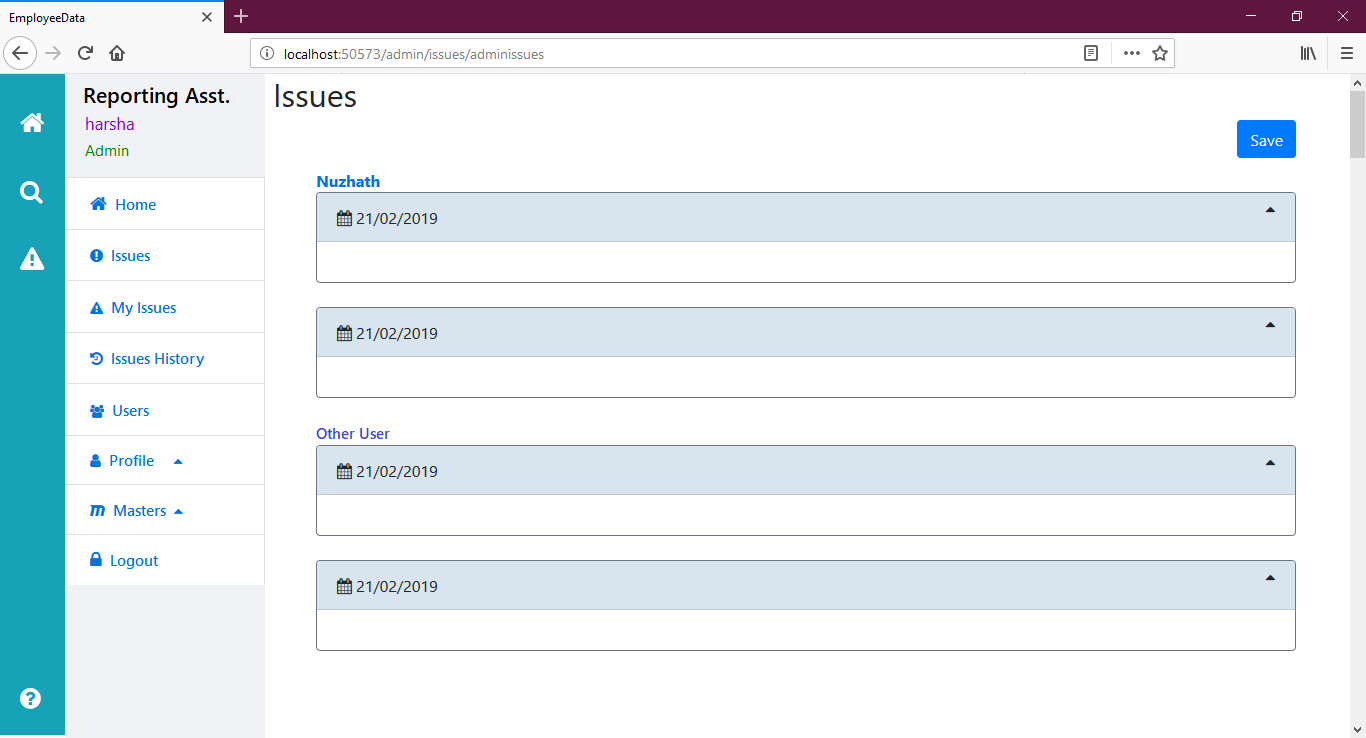
The admin can delete the project, only if it has no "tasks" / "tasks done" / "final comments".

**Screen 22: Admin - Search Projects Page**



Projects are to be displayed as the characters entered in the search box.

**Screen 23: Admin - Issues Page**



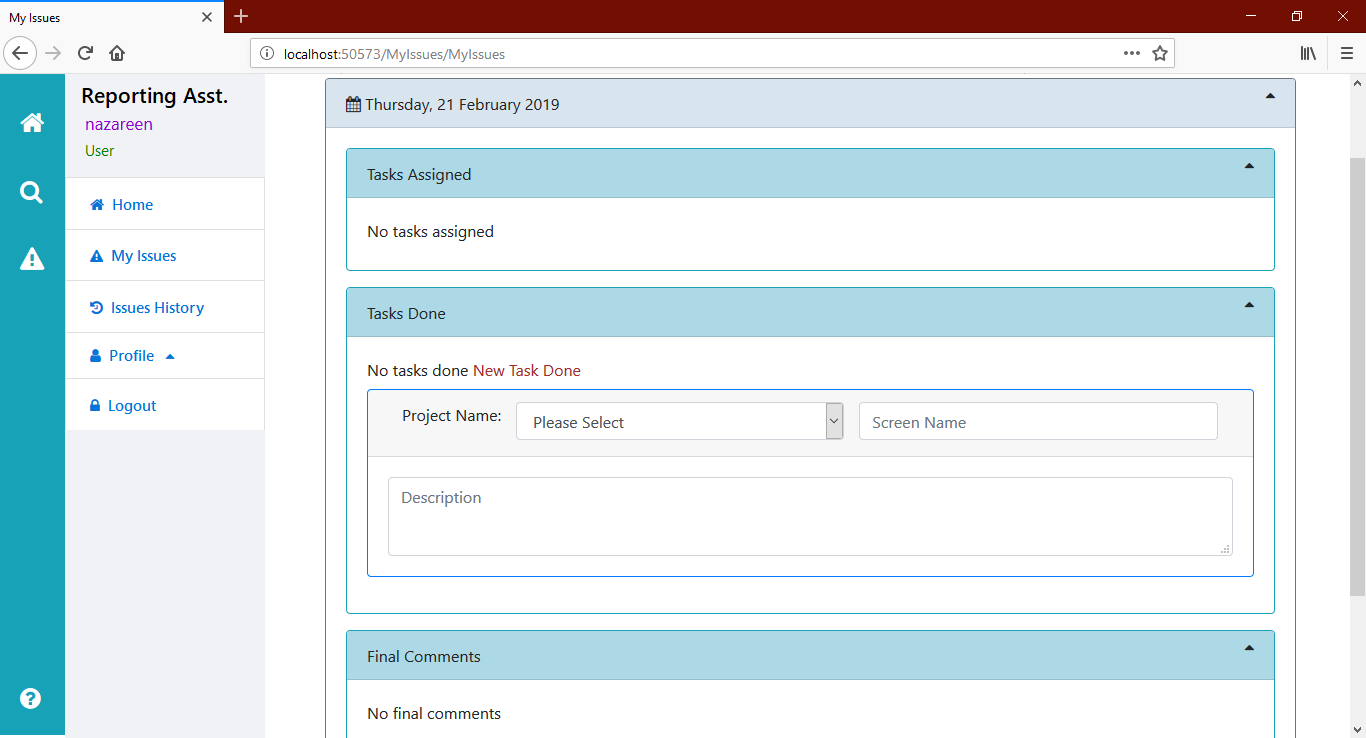
Admin can see all the user's tasks assigned / tasks done / final comments [given by any admin].

But admin can can edit "tasks assigned" / "final comments" of his own records only.

Admin can add "tasks assigned" and "final comments" to any user for any date.

In the "Issues History" page, same screen needs to be repeated for one week [instead of two dates].

**Screen 24: User - My Issues Page [if empty]**



If no records, it should display message like above.

**Additional Instructions:**

* Use HTML helpers
* Use Entity Framework Code First Approach with Code-First Migrations
* Use Strongly-Typed Views
* Use AntiForgeryToken
* Use Razor View Engine
* Use Layout view for "User" and another Layout view for "Admin"; with \_ViewStart.cshtml for each.
* Use Sorting, Paging with EF in Categories and Project pages.
* Create necessary validations as explained above.
* Use asp.net identity for registration, login, my profile, change password etc., pages.
* Create roles: "Admin" and "User" in Startup.cs file
* Create sample admins and users as specified above in Startup.cs file.
* There is no registration page for admin; admin can be created only through "Startup.cs" file.
* Use IAuthorizationFilter and IAuthenticationFilter
* Use Exception Handling using Custom Errors, IExceptionFilter
* Use Service Pattern along with Repository Pattern
* Use jQuery Validation Plugin, Bootstrap and Font-Awesome.
* It is optional to use AJAX [Web Api]

Please submit your source code after completing it, using CodePen.

**Questions for this assignment**

**Please submit your source code after completing it, using DotNetFiddle( https://dotnetfiddle.net/CsMvc ).**