iResolve Bug Tracker System

Project Description: - This project will take care of all day to day task of issue creation and resolution that comes in any phase (Development, Quality Assurance and Production) of any software products.

Modules: -

Login Module

Registration Module

Dashboard Module

Admin Dashboard

Project Manager Dashboard

Developer Dashboard

Tester Dashboard

Logout

Users: -

There will be 4 types of users of the system.

1. Admin
2. Project Manager
3. Developer (Tech Analyst, Tech Lead, Sr. Developer, Jr. Developer etc.)
4. Tester (Quality Assurance, Sr. Tester, Jr. Tester)

Application Flow: -

Header

1. You can create a beautiful header of the application which will contain “iResolve” (The name of the application).

Registration

1. New user can send a registration request. For it user has to provide.
2. First Name
3. Last Name
4. Email
5. Password

So the Registration Page will contain below fields

1. First Name (text html field)
2. Last Name (text html field)
3. Email (email html field)
4. Password (password html field)
5. Confirm Password (password html field)
6. There will be a Register button which will call API to register the user.
7. Initially the user will be Inactivated until admin logins to the system and assign a Role and Activate the user.
8. For our application we have four roles available namely ADMIN, PM, DEVELOPER and TESTER.
9. An ADMIN can assign any other user to ADMIN role.
10. Add a button “Active” to activate the user. Here you have to call API to activate the user. API will return success flag if the activation process completed successfully. If it is a success show message “Process completed successfully.” otherwise show “Process failed!”.

Login

1. Only Activated user can login to the system.
2. User will have to provide Email and Password in order to login successfully to the system.
3. Login Page will contain following field
4. Email (email html field)
5. Password (password html field)
6. Add one Login Button.
7. Call Login API in order to validate the Credentials.
8. API will return response containing information related the authenticity of the user along with the role assigned to the user.
9. If there is any error in the credentials. Show message to the user “Either Email or Password is incorrect.
10. If Credentials is correct but the user is not activated by the Admin. Show message to the user “You are not an active user. Please contact to the Administrator.”

Dashboard

1. After successful login, user will be navigated to the respective Dashboard.
2. Every type of user (based on user’s role) will have different dashboard based on what the user is authorized to do.
3. So there will be 4 Dashboards (Admin Dashboard, Project Manager Dashboard, Developer Dashboard and Tester Dashboard).

Logout

1. Add one Logout (link or button) in the header of the application.

Functionality based on User Role

ADMIN

Functions has been written in Registration section.

Dashboard

1. Show all the Inactivated user(s) in a grid or table as soon as the dashboard loads inside the application. In order to do this call API in ngOnInit() hook of the component.

1. Grid or Table will contain following columns.
2. First Name (read only field)
3. Last Name (read only field)
4. Email (read only field)
5. Role (dropdown)
6. Activate (button)
7. Role dropdown in the grid will have all the roles available in the application. In order to load it you have to call the API.
8. Once Admin will click on Activate button, another API will be called which will take the user information and update the record in the database.

PM

1. A Project Manager can create a new Project.
2. It can also assign or unassign Developer(s) and Tester(s) to a project.

Dashboard

1. Add one button “New Button” to create a new project.
2. New Project form will take following fields:
3. Project Name (text field)
4. Project Description (text area)
5. Create (button)
6. Click on Create button will call an API to create a new project.
7. Show projects created (old or new) by the logged in Project manager on the dashboard (in card layout).
8. Upon clicking the project card, show the project details.
9. Every card will contain two buttons: -
10. Assign Developer(s)
11. Assign Tester(s)
12. Clicking on the Assign Developer(s) button will call an API, API will return list of all developers. Open a new pane or section which will have a grid containing the following columns:
13. First Name (read only)
14. Last Name (read only)
15. Role (read only)
16. Action (check box)
17. Add a Save button below which will call an API to Assign the checked Developers to the project.
18. Clicking on the Assign Tester(s) button will call an API, API will return list of all testers. Open a new pane or section which will have a grid containing the following columns:
19. First Name (read only)
20. Last Name (read only)
21. Role (read only)
22. Action (check box)
23. Add a Save button below which will call an API to Assign the checked Testers to the project.
24. Show success or failure message in each operation.

DEVELOPER

1. Developer can work on the assigned issue(s).

Dashboard

1. Show all the assigned issue(s) to the developer on the dashboard in a grid or table.
2. You have to call an API in the ngOnInit() hook which will give you the list of all assigned issue(s) to the logged in user.
3. Grid or table will contain below columns: -
4. Issue Id (read only)
5. Title (read only)
6. Description (read only)
7. Project Name (read only)
8. Priority (read only)
9. Severity (read only)
10. Status (read only)
11. Action

1. Title of the Issue will be a link. This link will open the details of the Issue in a different page in edit mode.
2. Add a button “Edit” or icon under Action column to edit the issue.
3. Both Title link and Edit button will do the same thing.
4. In the edit mode, the form will contain the following fields:
5. Issue ID (read only, text field)
6. Title (read only, text field)
7. Description (read only, text area)
8. Project Name (read only, text field)
9. Discussion (editable, text area)
10. Priority (editable, dropdown)
11. Severity (editable, dropdown)
12. Status (editable, dropdown)
13. Created By (read only, tester name)
14. Created On (read only, date field)
15. You have to call API to get all priorities, severities and status available in the application.
16. It will also contain a button “Save” which will be used to save the information edited by the developer. You have to call an API to save the information.

TESTER

1. A tester can create a new issue as well as update an old issue.

Dashboard

1. Show all the assigned issue(s) to the developer on the dashboard in a grid or table.
2. You have to call an API in the ngOnInit() hook which will give you the list of all assigned issue(s) to the logged in user.
3. Grid or table will contain below columns: -
4. Issue Id (read only)
5. Title (read only)
6. Description (read only)
7. Project Name (read only)
8. Priority (read only)
9. Severity (read only)
10. Status (read only)
11. Action
12. Title of the Issue will be a link. This link will open the details of the Issue in a different page in edit mode.
13. Add a button “Edit” or icon under Action column to edit the issue.
14. Both Title link and Edit button will do the same thing.
15. In the edit mode, the form will contain the following fields:
16. Issue ID (read only, auto generated in database)
17. Title (editable, text field)
18. Description (editable, text area)
19. Project Name (editable, text field)
20. Discussion (editable, text area)
21. Priority (editable, dropdown)
22. Severity (editable, dropdown)
23. Status (editable, dropdown)
24. Created On (read only, date field, current date)
25. You have to call API to get all priorities, severities and status available in the application.
26. It will also contain a button “Save” which will be used to save the information edited by the developer. You have to call an API to save the information.
27. Add a new button “New Issue” to create a new Issue. A form will be open to fill all the details about the new issue. The form will be exactly same what we have in the edit mode mentioned in serial number 7. Remember – A tester can edit most of the information.
28. An API will be called to save the information in the database.
29. Show success or failure message on each operation.

Note: -

1. This document can be updated if it is required. Any change of this document will come into a different version. Current version of this document is v1. Next version will be v2 so on and so forth.
2. We call any API only when we have to interact with the database, either we have to get some data from the database or we have to store any data to the database.
3. I have created REST API. You have to call this API using only three API methods:-
4. GET - When you are getting any information from the database. For example- role list or issue list
5. POST - When you create a new record in the database. For example – While creating a new Project or Issue
6. PUT - When you update the record present inside database. For example- While updating an Issue by developer or a tester.
7. API related information will be given in a different document.