

TASK 1: REQUIREMENT

1.1 Use Case Diagram

1.1.1 Phase 1

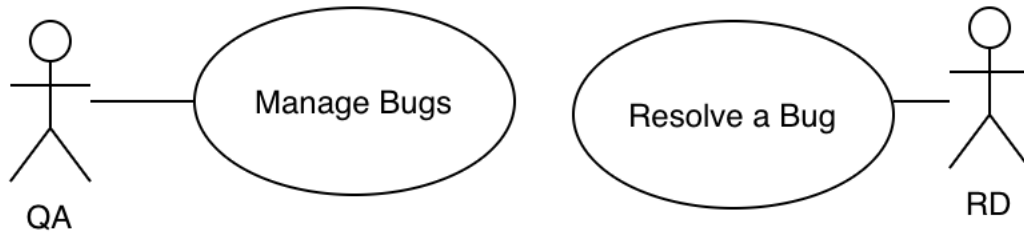


Figure 1.1: Use Case Diagram for Phase 1

1.1.2 Phase 2

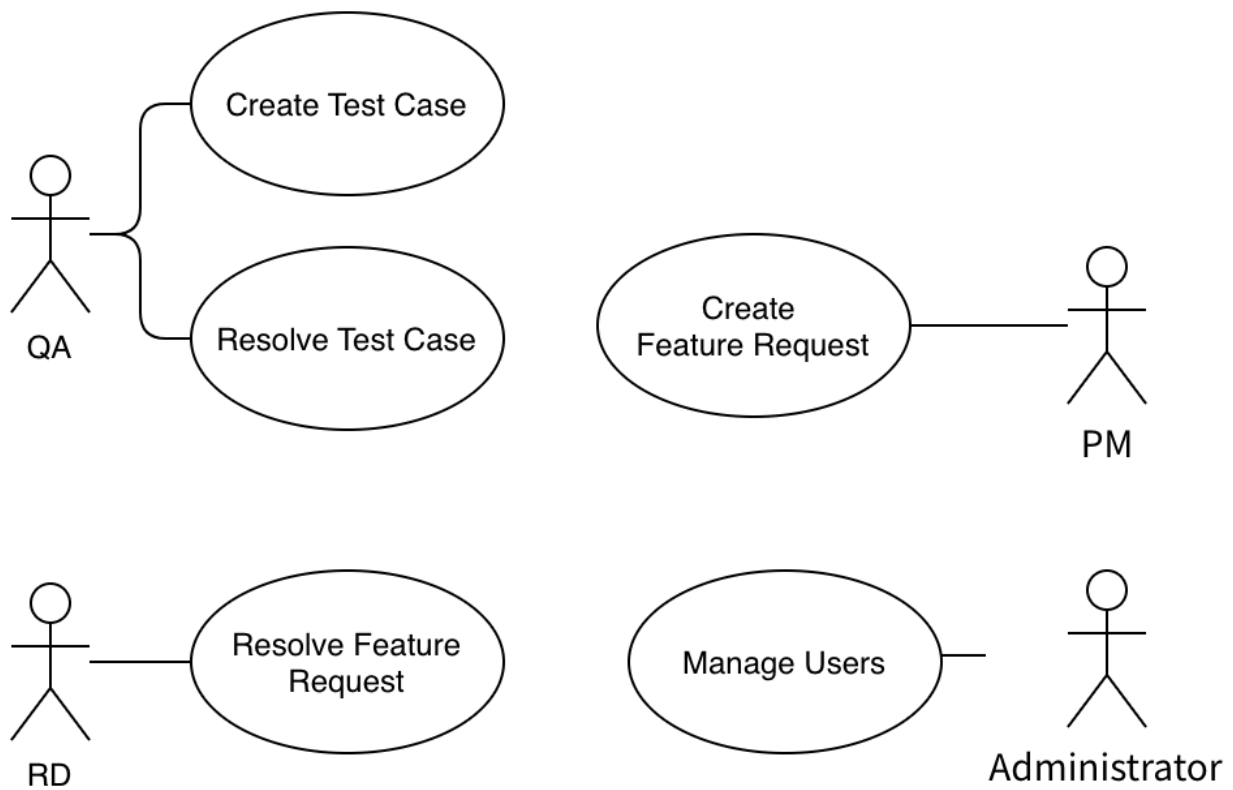


Figure 1.2: Use Case Diagram for Phase 2

1.2 Expanded Use Case

1.2.1 Phase 1

Table 1.1: Expanded Use Case for Manage Bugs

Use Case 1	Manage Bugs	
Goal in Context	To allow QA to report a bug in the system	
Primary Actor	QA	
Trigger	QA identifies a bug during testing	
Typical Course of Events		
Actor Action		System Response
1. This use case begins when the QA identifies a bug during testing		
2. The QA logs into the system		The system displays the login screen and validates the provided information
3. The QA selects the "Create Bug" option		The system displays the bug creation form
4. The QA provides a summary and description of the bug		System records the provided information. Bug is created and stored in the database. A bugID is auto generated and status is set to OPEN
Alternative Course of Events		
Line 3a: If merchant wants to edit a bug, the QA chooses to edit a bug. The QA selects bug to edit. QA modifies summary or description. The system saves changes		
Line 3b: If merchant wants to delete an existing bug, the QA will click on the Delete button next to the record. The system displays a confirmation box before deleting the record from the database		
Line 4: If the QA did not fill all required data correctly, the system will prompt the QA to re-enter again		

Table 1.2: Expanded Use Case for Resolves Bug

Use Case 2	Resolves Bug
------------	--------------

Goal in Context	To allow RD to resolve bugs in the system
Primary Actor	RD
Trigger	RD selects the "Resolve Bug" option
Typical Course of Events	
Actor Action	System Response
1. This use case begins when the RD selects the "Resolve Bug" option	
2. The RD logs into the system	The system displays the login screen and validates the provided information
3. The RD selects "Resolve Bug" option	The system displays a list of unresolved bugs for RD to choose from
4. The RD selects a bug to resolve	The system marks the bug as resolved. Status is updated to RESOLVED
Alternative Course of Events	
-	

1.2.2 Phase 2

Table 1.3: Expanded Use Case for Creates Test Case

Use Case 1	Creates Test Case
Goal in Context	To allow QA to create a test case
Primary Actor	QA
Trigger	QA decides to create a new test case
Typical Course of Events	
Actor Action	System Response
1. This use case begins when the QA decides to create a new test case	
2. The QA logs into the ticket tracking system	The system displays the login screen and validates the provided information

3. The QA selects the "Create Test Case" option	The system displays the form for creating a new test case
4. The QA enters a required data of the test case	The system saves the record into database after validating all user inputs. A testCaseId is auto generated
5. All users can view the test case	The system displays the test cases list
Alternative Course of Events	
-	

Table 1.4: Expanded Use Case for Resolve Test Case

Use Case 2	Resolve Test Case	
Goal in Context	To allow QA to mark a test case as resolved.	
Primary Actor	QA	
Trigger	QA wants to mark a test case as resolved after successful testing.	
Typical Course of Events		
Actor Action	System Response	
1. This use case begins when the QA decides to create a new test case		
2. The QA logs into the ticket tracking system	The system displays the login screen and validates the provided information	
3. The QA navigates to the list of test cases	The system display the unresolved test cases list	
4. The QA selects the specific test case to be marked as resolved	The system updates the status of the test case to "RESOLVED"	
6. All users can view the test case	The system displays the test cases list	
Alternative Course of Events		

-

Table 1.5: Expanded Use Case for Creates Feature Request

Use Case 3	Creates Feature Request	
Goal in Context	To allow the PM to create a new feature request in the ticket tracking system	
Primary Actor	PM	
Trigger	PM decides to create a new feature request	
Typical Course of Events		
Actor Action		System Response
1. This use case begins when the PM decides to create a new feature request		
2. The PM logs into the ticket tracking system		The system displays the login screen and validates the provided information
3. The PM navigates to the "Create Feature Request"		The system displays the form for creating a new feature request
4. The PM provides a required data for the new feature request. The PM sets the severity and priority of the feature request (if applicable)		The system creates a new feature request in the database. The featureId is auto generated. The system sets the status of the feature request to "Open"
Alternative Course of Events		
Line 4: If the PM did not fill all required data correctly, the system will prompt the PM to re-enter again		

Table 1.6: Expanded Use Case for Resolves Feature Request

Use Case 4	Resolves Feature Request
Goal in Context	To allow the RD to mark a feature request as resolved
Primary Actor	RD
Trigger	RD decides to resolve a feature request

Typical Course of Events	
Actor Action	System Response
1. This use case begins when the RD decides to resolve a feature request	
2. The RD logs into the ticket tracking system	The system displays the login screen and validates the provided information
3. The RD navigates to the list of feature requests	The system displays a list of feature requests available to RD
4. The RD selects a specific feature request	The system displays detailed information about the selected feature request
5. The RD marks the feature request as resolved	System updates the status of the feature request to "RESOLVED"
Alternative Course of Events	
-	

Table 1.7: Expanded Use Case for Manages Users

Use Case 5	Manages Users
Goal in Context	To allow the Administrator to manage QA, RD, or PM users in the system
Primary Actor	Administrator
Trigger	Administrator initiates the user management process
Typical Course of Events	
Actor Action	System Response
1. This use case begins when the Administrator initiates the user management process	
2. The Administrator logs into the ticket tracking system	The system authenticates the Administrator
3. The Administrator navigates to the user management section	The system displays the user management interface

4. The Administrator selects the option to add a new user	The system displays a form for adding a new user
5. The Administrator fills in the required information (username, password, role, etc.) for the new user	The system validates the information and creates the new user
Alternative Course of Events	
Line 4a: The Administrator may choose to modify existing users (change roles, update information). The system updates the user information as per Administrator's modifications	
Line 4b: The Administrator may choose to delete a user. The system prompts for confirmation and deletes the user upon confirmation	
Line 5: If the entered user information is invalid. The system displays an error message and prompts the Administrator to correct the information	

1.3 Class Diagram

1.3.1 Phase 1

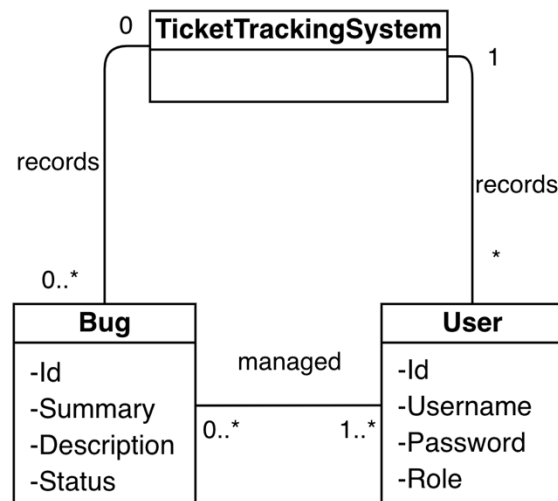


Figure 1.3: Class Diagram for Phase 1

TASK 2: IMPLEMENTATION

The implementation focused on utilizing .NET Core MVC to meet the requirements, resulting in a minimalistic user interface. The result of the implementation is all of the requirements have been successfully completed.

2.1 Tech stacks

Razor, .NET CORE MVC, MongoDB

2.2 GitHub Project Link

<https://github.com/suryapradipta/TicketTrackingSystem>

2.3 Default Accounts

Username: adminqa

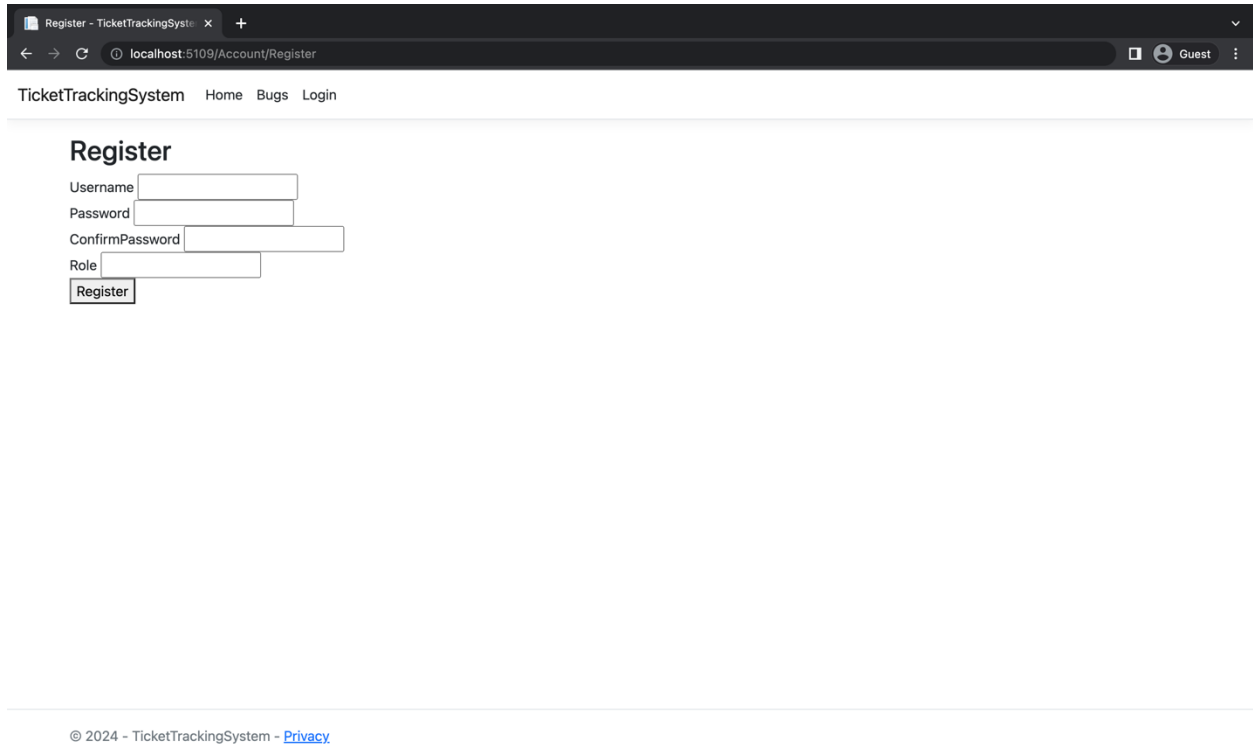
Password: adminqa

Username: adminrd

Password: adminrd

2.4 User Interface

2.4.1 Register Account



The screenshot shows a web browser window with the title "Register - TicketTrackingSystem". The address bar shows "localhost:5109/Account/Register". The page has a navigation bar with "TicketTrackingSystem", "Home", "Bugs", and "Login". The main content area is titled "Register" and contains a form with the following fields: "Username", "Password", "ConfirmPassword", and "Role". Below the "Role" field is a "Register" button. The footer shows "© 2024 - TicketTrackingSystem - [Privacy](#)".

Register - TicketTrackingSystem

localhost:5109/Account/Register

TicketTrackingSystem Home Bugs Login

Register

Username

Password

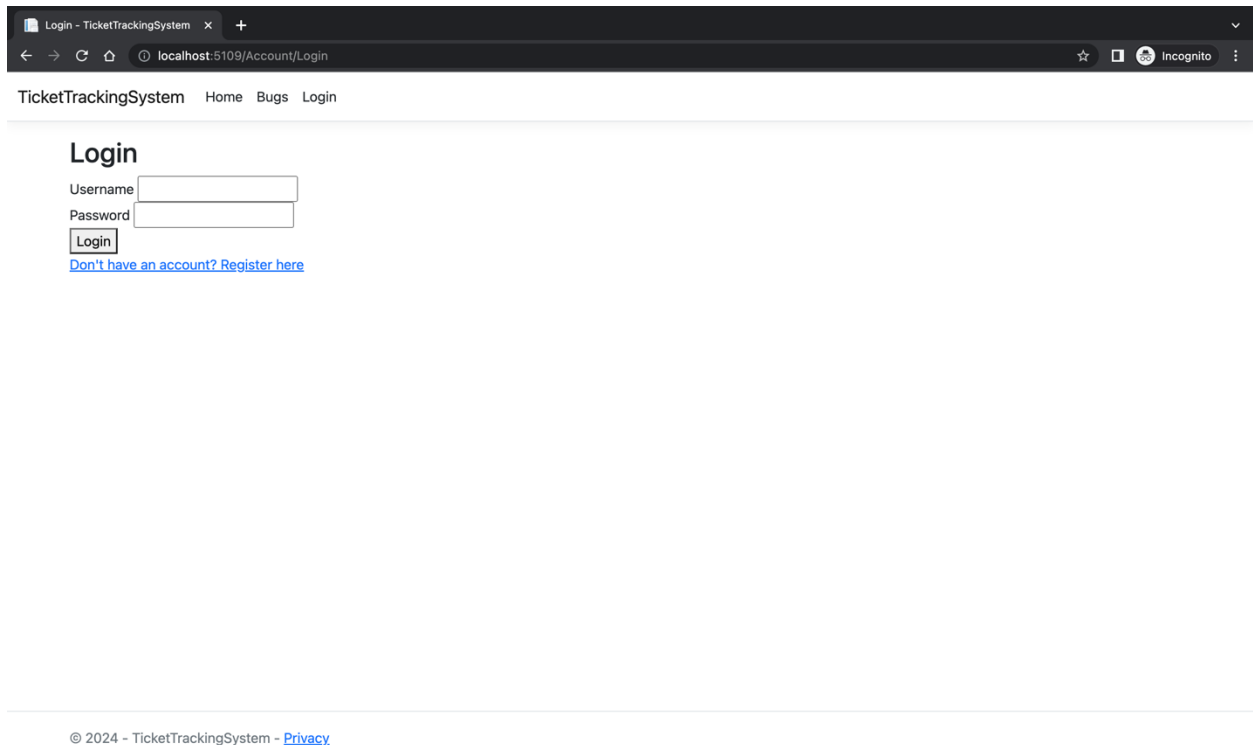
ConfirmPassword

Role

Register

© 2024 - TicketTrackingSystem - [Privacy](#)

2.4.2 Login



The screenshot shows a web browser window with the title "Login - TicketTrackingSystem". The address bar shows "localhost:5109/Account/Login". The page has a navigation bar with "TicketTrackingSystem", "Home", "Bugs", and "Login". The main content area is titled "Login" and contains a form with the following fields: "Username" and "Password". Below the "Password" field is a "Login" button. Below the "Login" button is a link: "Don't have an account? [Register here](#)". The footer shows "© 2024 - TicketTrackingSystem - [Privacy](#)".

Login - TicketTrackingSystem

localhost:5109/Account/Login

TicketTrackingSystem Home Bugs Login

Login

Username

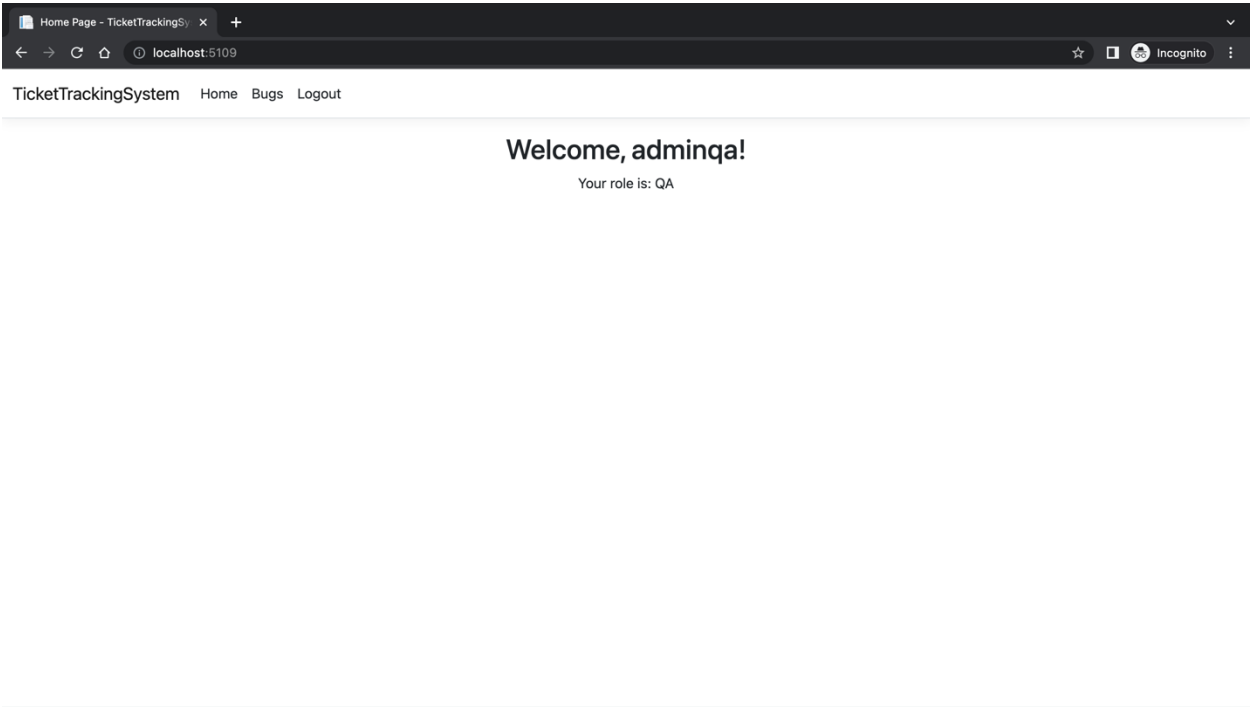
Password

Login

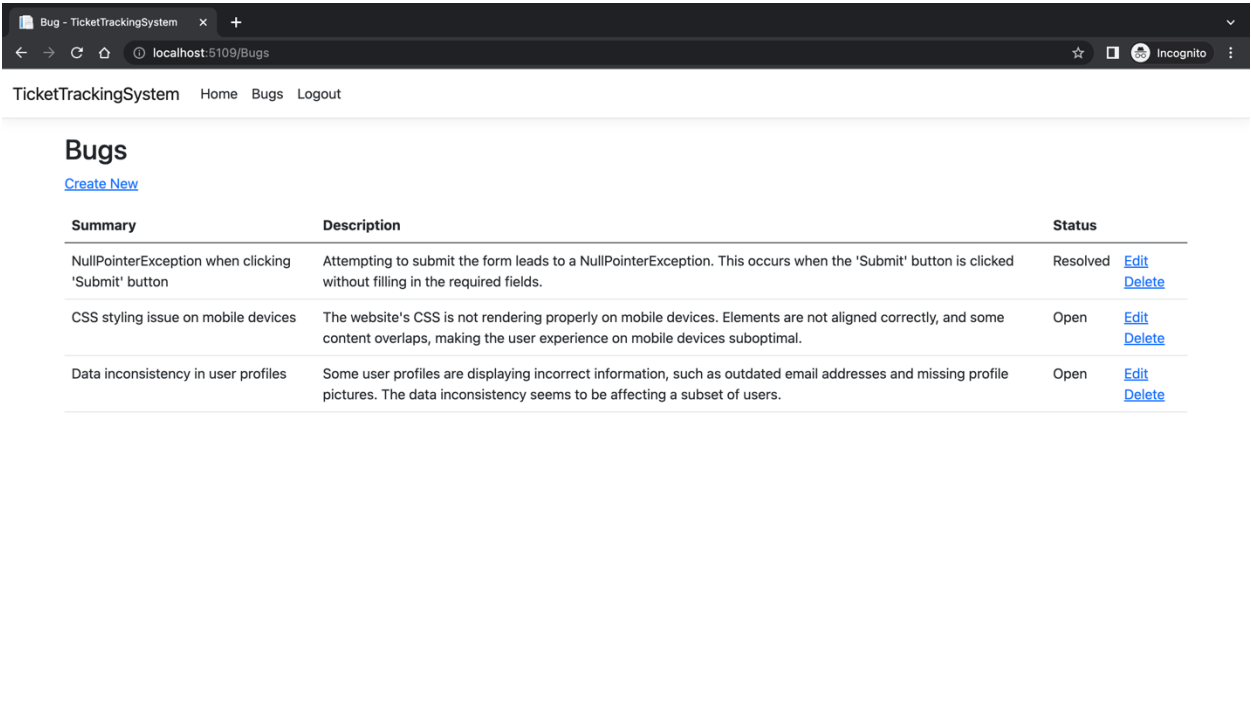
[Don't have an account? Register here](#)

© 2024 - TicketTrackingSystem - [Privacy](#)

2.4.3 Dashboard



2.4.4 Manage Bugs by QA



2.4.5 Create Bug

Create - TicketTrackingSystem

localhost:5109/Bugs/Create

Incognito

[TicketTrackingSystem](#) [Home](#) [Bugs](#) [Logout](#)

Create

Summary

Description

Create

[Back to List](#)

© 2024 - TicketTrackingSystem - [Privacy](#)

2.4.6 Edit Bug

Edit - TicketTrackingSystem

localhost:5109/Bugs/Edit/65a61966680382373b343782

Incognito

[TicketTrackingSystem](#) [Home](#) [Bugs](#) [Logout](#)

Edit

Summary

Description

Status

Open

Save

[Back to List](#)

© 2024 - TicketTrackingSystem - [Privacy](#)

2.4.7 Delete Bug

Delete - TicketTrackingSystem

localhost:5109/Bugs/Delete/65a61966680382373b343782

Incognito

[TicketTrackingSystem](#) [Home](#) [Bugs](#) [Logout](#)

Delete

Are you sure you want to delete this?

Bug

Summary

CSS styling issue on mobile devices

Description

The website's CSS is not rendering properly on mobile devices. Elements are not aligned correctly, and some content overlaps, making the user experience on mobile devices suboptimal.

Status

Open

Delete

[Back to List](#)

© 2024 - TicketTrackingSystem - [Privacy](#)

2.4.8 Manage Bugs by RD

Bug - TicketTrackingSystem

localhost:5109/Bugs

Incognito

[TicketTrackingSystem](#) [Home](#) [Bugs](#) [Logout](#)

Bugs

Summary	Description	Status
NullPointerException when clicking 'Submit' button	Attempting to submit the form leads to a NullPointerException. This occurs when the 'Submit' button is clicked without filling in the required fields.	Resolved
CSS styling issue on mobile devices	The website's CSS is not rendering properly on mobile devices. Elements are not aligned correctly, and some content overlaps, making the user experience on mobile devices suboptimal.	Open Resolve
Data inconsistency in user profiles	Some user profiles are displaying incorrect information, such as outdated email addresses and missing profile pictures. The data inconsistency seems to be affecting a subset of users.	Open Resolve

© 2024 - TicketTrackingSystem - [Privacy](#)

2.4.9 Resolve Bug

Resolve - TicketTrackingSystem

localhost:5109/Bugs/Resolve/65a61959680382373b343781

Incognito

TicketTrackingSystem

HomeBugsLogout

Resolve

Are you sure you want to resolve this bug?

Bug

Summary

NullPointerException when clicking 'Submit' button

Description

Attempting to submit the form leads to a NullPointerException. This occurs when the 'Submit' button is clicked without filling in the required fields.

Status

Open

Resolve

Back to List

© 2024 - TicketTrackingSystem - [Privacy](#)