**DEFECT TRACKING SYSTEM HLD V1**

|  |  |
| --- | --- |
| Application Name | Defect Tracking |
| Author | Devendra Kumar Mishra |

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Comments** | **Author** |
| 1.0.0 |  |  | Gourav Makwana |
|  |  |  |  |

Copyright

“This document is confidential to Rakuten. The information contained herein is not to be distributed, revealed or disseminated outside the company to any other party without the prior expressed consent and written permission of Rakuten.”

Table of Contents

[Table of Contents 3](#_Toc91777301)

[1 Introduction 4](#_Toc91777302)

[2 Service Definition 5](#_Toc91777303)

[3 Architecture (Solution/Software) 6](#_Toc91777304)

[**3.1** **Architecture diagram** 6](#_Toc91777305)

**3.2** **ER**-**Diagram**…………………………………………………………………………………………………………………………………7

[**3.3** **Security** 10](#_Toc91777308)

[**3.4** **Authentication & Authorization** 11](#_Toc91777309)

[4 User Interface 12](#_Toc91777313)

[5 Reference 20](#_Toc91777314)

## Introduction

Its purpose is to create a track of defect and to generate a ticket for the defect that found in project. The ticket is generated by the QA and at the time of generating the ticket QA assign the ticket to the developer for fixing the defect. When the testers encounter ‘n’ no. of defects, he generates a unique ticket for each individual defect in a single project. The Ticket information along with its status, remark and assign\_to\_id in project and show on developer dashboard to whose assign the ticket. This is “Ticket”. These are stored in the database. This is useful for further reference. Ticket information includes the ticket id, ticket name, ticket status, project name, assign\_to\_id, ticket description. This whole process continues until ticket are got fixed in the project.

The ticket report is show to the QA and the developer dashboard as ticket is generated by the QA. This makes that no error will go unfixed because of poor communication. Defect Tracking System plays a vital role in the testing phase. It supports assigning ticket for the developer, project by the QA. The Defect Tracking System maintains the different users separately i.e., it provides separate environments for admin, developer and QA.

The main objectives of the Defect Tracking System are:

* QA Identifying the Defects in the developed application.
* No Defect will be unfixed in the developed application.
* Not merely identifying the Defects but also providing the Defect information
* As soon as the Defects are identified. They are reported to developer.
* To ensure that who needs to know about the Defect can learn soon after it is reported

## Service Definition

Major objective of this Defect Tracking system application was to provide a platform for track the defect buy QA and developer from the website. The application consists of three type of user –

* Admin – A admin can log in to the application using username and password. On successful login he will be redirected to admin-dashboard where he will see an overall view of the application which includes – User listing, add user, update user, project listing, add project, update project.
  + Admin can view all the user and can add the user and can also edit them.
  + Admin can view all the project and then change the project description and can also add the project.
  + Admin can provide the username and password to the user that registered by the admin.
* QA – A QA can log in to the application using username and password that provide by the admin. On Successful login he will be redirected to the QA -dashboard where he will see the application service that is provided to the QA which includes- project listing, ticket listing, add ticket, edit ticket.
* QA can view all the project that are created by the admin.
* QA can view all the ticket that are generated in the project by themselves or by other QA.
* QA can add new ticket in the project and assign it to the developer and can also edit the ticket.
* Developer – A developer can log in to the application using username and password that provide by the admin. On Successful login he will be redirected to the developer-dashboard where he will see the application service that is provided to the developer which include- project listing, ticket list, and edit ticket.
* Developer can view all the project that are created by the admin.
* Developer can view all the ticket that are generated in the project by the QA and also he can see ticket that are assign to it separately.
* Developer can edit ticket in the project that assign to it by the QA.

## Architecture (Solution/Software)

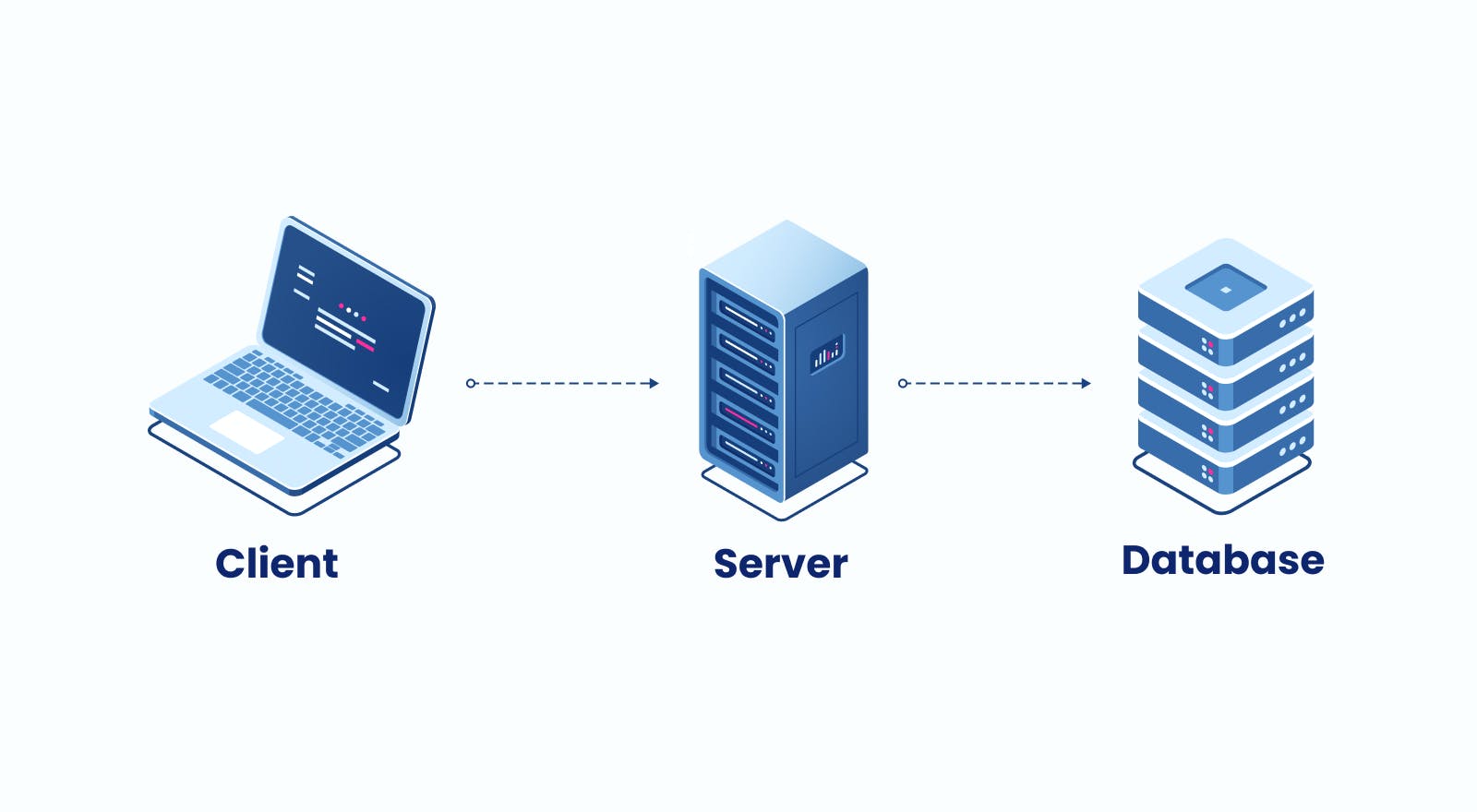
### 3.1 Architecture Design

The overall software can be divided into three parts –

1. Frontend – This side of the application resides in the client part. The user will use the UI of our application to access the features/functionality. The Frontend is built using the Angular Framework.

Angular is an application design framework and development platform for creating efficient and sophisticated single-page apps. With the help of Angular we can create component based scalable web application. Angular comes with a bunch of libraries that covers a wide range of features including routing, forms management, client-server communication and much more.

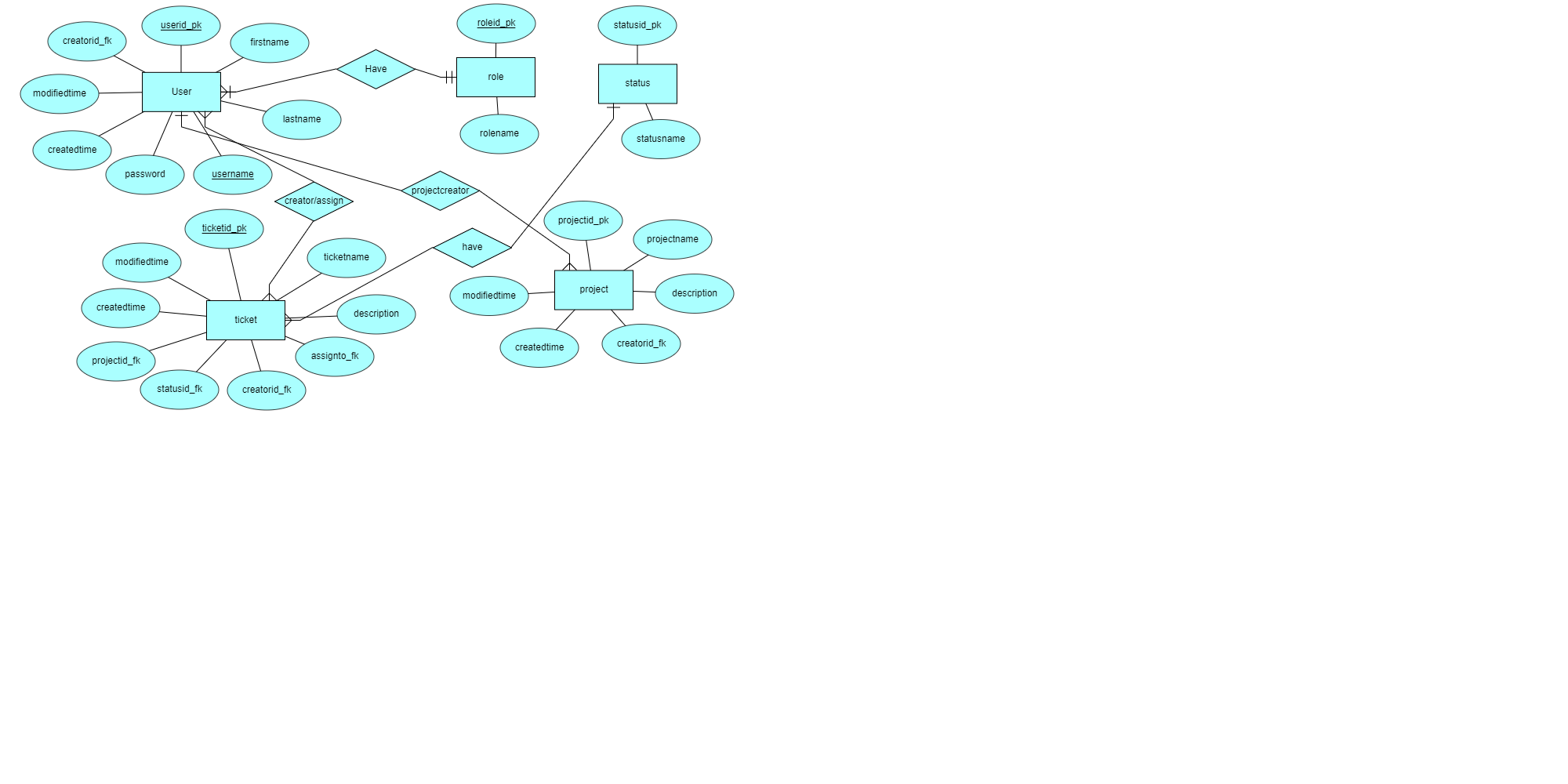
1. Backend – The Frontend part uses the api served by the backend which was built using spring boot framework. Spring boot is a project that is built on the top of spring framework. It provides an easier and faster way to set-up, configure and run a REST API.



High level diagram of Defect Tracking

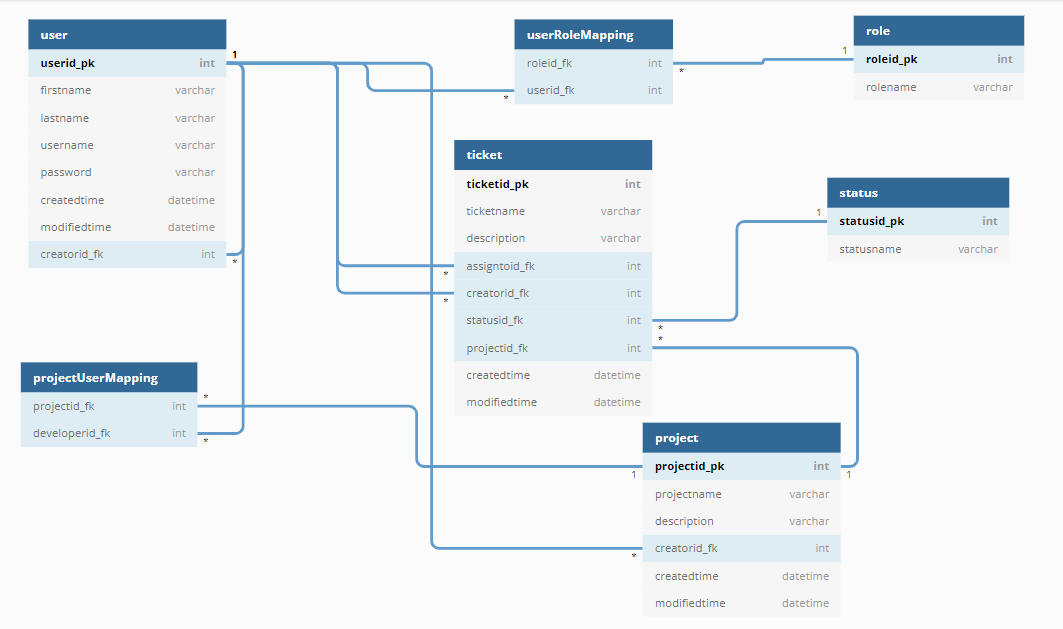
1. Database – MySQL is used as database management system in our application. MySQL is an open source relational database management system. MySQL is ideal for both small- and large-scale applications.

### ER Diagram



The overall Defect Tracking project is backed by 7entities and their multiple attributes, we will see each entity one by one –

1. User – This is the most crucial entity which is used authentication and authorization. User entity include 8 attributes which are – userid\_pk, firstname, lastname, username, password, createdtime, modifiedtime and creatorid. Here creatorid is a type User which will tell that the user is created by which user.
2. Role - This is the second entity in the database. It include 2 attributes which are roleid\_pk and rolename.
3. UserRoleMapping – This entity defines the role of the user that the user have. It include 2 attributes which are role\_id\_fk and user\_id\_fk.
4. Project – The project entity is the most useful entity. It include 6 attribute which are – projecteid\_pk, projectname, description, creatorid, createdtime and modifiedtime.
5. ProjectUserMapping – This entity defines the project that are created by the user. It defines which project is created by the same user. It includes 2 attributes which are – projecteid\_fk and developerid\_fk.
6. Status – This entity is used to show the status of the ticket. It include 2 attributes which are – statusid\_pk and statusname.
7. Ticket – The ticket entity is used to store the ticket created by the user. It includes 9 attributes which are – ticketid\_pk, ticketname, description, assigntoid\_fk, creatorid\_fk, createdtime, modifiedtime, statusid\_fk and projected\_fk.

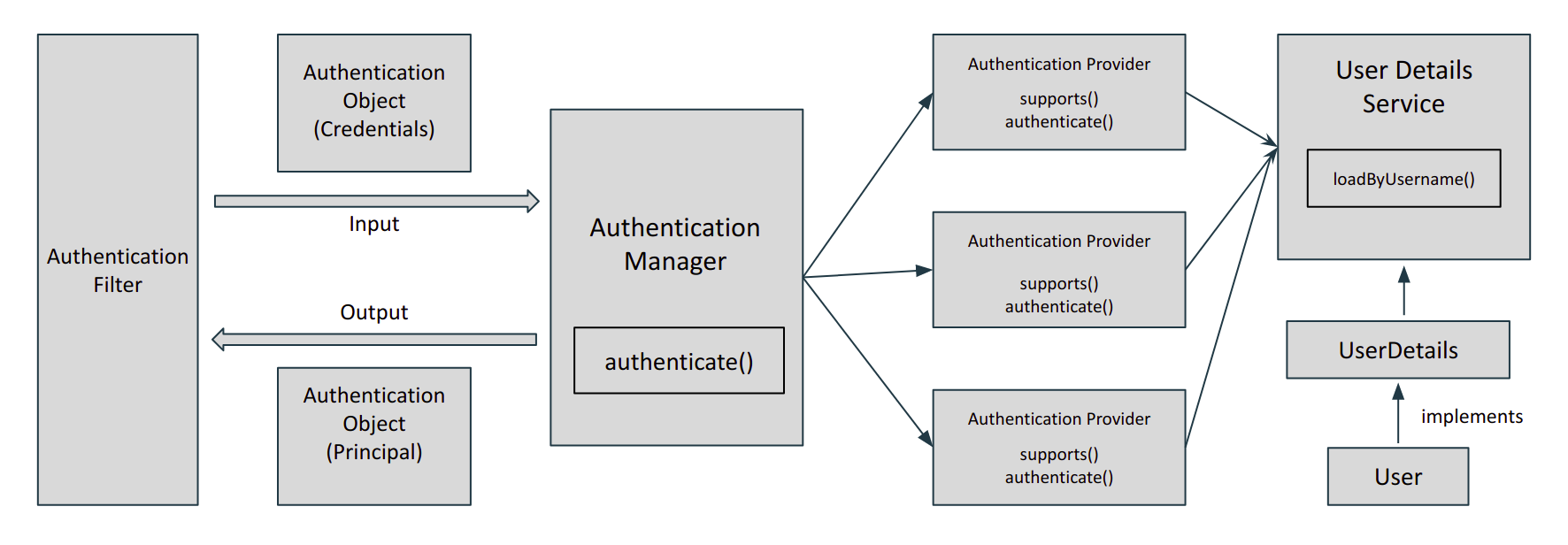
**Database Design**

Above diagram represents the entire database design of our Defect Tracking Web Application. Following is the detailed explanation of all the tables –

1. User Table
   1. Userid\_pk – it is the auto increment integer type primary key of user table.
   2. firstname – stores the firstname of the user.
   3. lastname – stores the lastname of the user.
   4. Username – stores the username of a user, which will be unique.
   5. Password – stores the password of a user in encrypted format
   6. createdtime – stores the createdtime of the user
   7. modifiedtime – stores the modifiedtime of the user.
   8. Creatorid\_fk - it is User type stores the creator of that user.
2. Role
3. Roleid\_pk – it is the auto increment integer type primary key of the role table.
4. Rolename – it is the varchar type stores the rolename of the role.
5. UserRoleMapping
6. Roleid\_fk– it is a foreign key which references to the roleid column of role table.which tell role of the user.
7. Userid\_fk - it is a foreign key which references to the userid column of user table.
8. Project
9. Projectid\_pk – it is the auto increment integer type primary key of the project table.
10. Projectname – it stores the projectname of the project.
11. Description – it stores the description of the project.
12. Creatorid\_fk - it is a foreign key which references to the userid column of user table.
13. createdtime – stores the createdtime of the project
14. modifiedtime – stores the modifiedtime of the project.
15. ProjectUserMapping
16. Projectedid\_fk - it is a foreign key which references to the projectid column of project table.
17. Developerid\_fk - it is a foreign key which references to the userid column of user table.
18. Status
19. Statusid\_pk – it is the auto increment integer type primary key of the status table.
20. Statusname – it stores the statusname of the status.
21. Ticket
22. Ticketid\_pk – it is the auto increment integer type primary key of the ticket table.
23. Ticketname – it stores the ticketname of the ticket.
24. Description – it stores the description of the ticket.
25. Assigntoid\_fk – it is a foreign key which references to the userid column of user table.
26. Creatorid\_fk – it is a foreign key which references to the userid column of user table.
27. Statusid\_fk – it is a foreign key which references to the statusid column of status table
28. Projectid\_fk – it is a foreign key which references to the projectid column of project table
29. createdtime – stores the createdtime of the project
30. modifiedtime – stores the modifiedtime of the project.

**Frontend** **Design**

3.3 Security

For the security, we have used spring-security framework. Spring security is a powerful and highly customizable authentication and access control framework. It is a de-facto standard for spring-based application. Following is the internal working of spring security.

Here is how spring security works,

When we add spring security starter jar, it internally adds filter to the application. A filter is an object that is invoked at the pre-processing and the post-processing of a request. It can manipulate a request or even stop it from reaching a servlet. There are multiple filters in spring security out of which one is the Authentication Filter which initiates the process of authentication.

Once the request passes through the authentication filter, the credentials of the user are stored in the Authentication object. Now, what actually is responsible for authentication is AuthenticationProvider (Interface that has method authenticate()). A spring app can have multiple authentication providers, one may be using OAuth, others may be using LDAP. To manage all of them, there is an AuthenticationManager.

The authentication manager finds the appropriate authentication provider by calling the supports() method of each authentication provider. The supports() method returns a boolean value. If true is returned, then the authentication manager calls its authenticate() method. After the credentials are passed to the authentication provider, it looks for the existing user in the system by UserDetailsService. It returns a UserDetails instance which the authentication provider verifies and authenticates. If success, the Authentication object is returned with the Principal and Authorities otherwise AuthenticationException is thrown.

###### 3.4 Authentication & Authorization

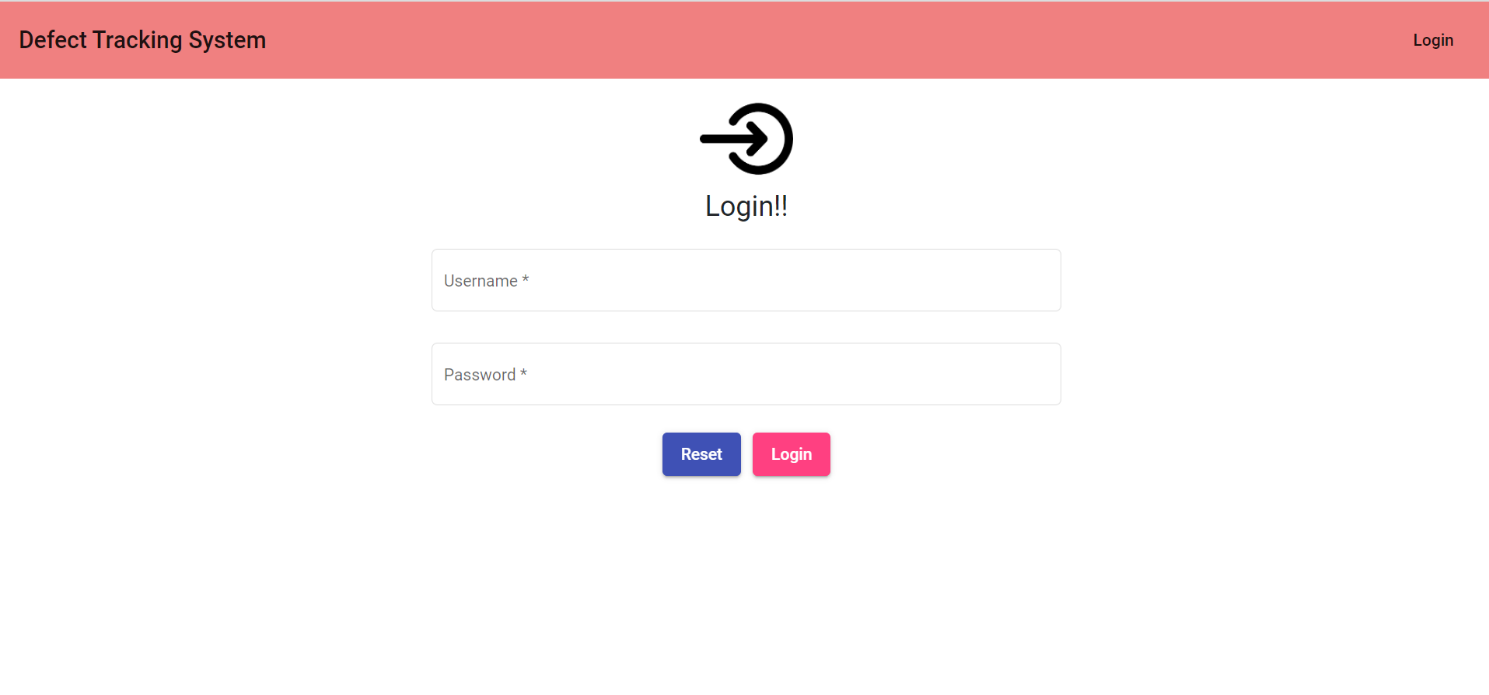
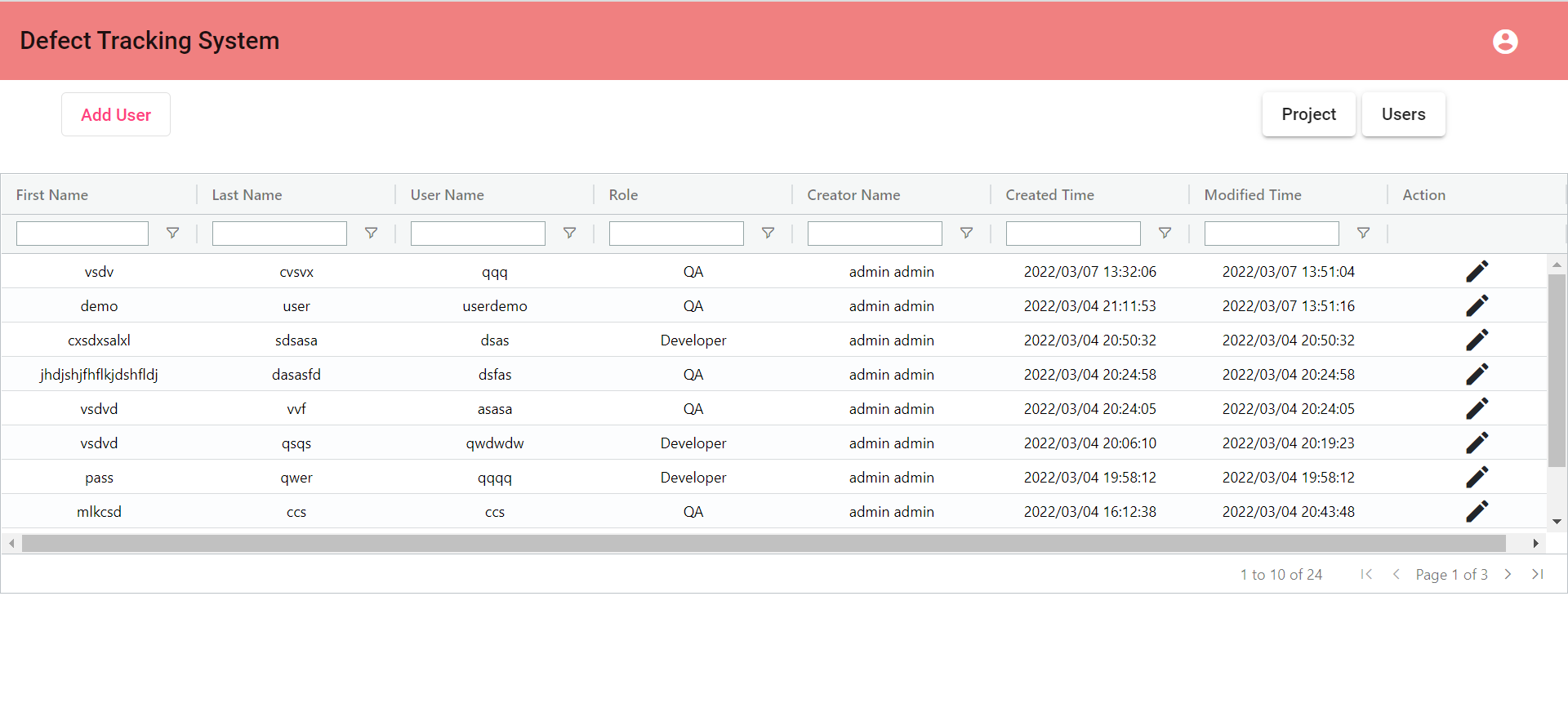
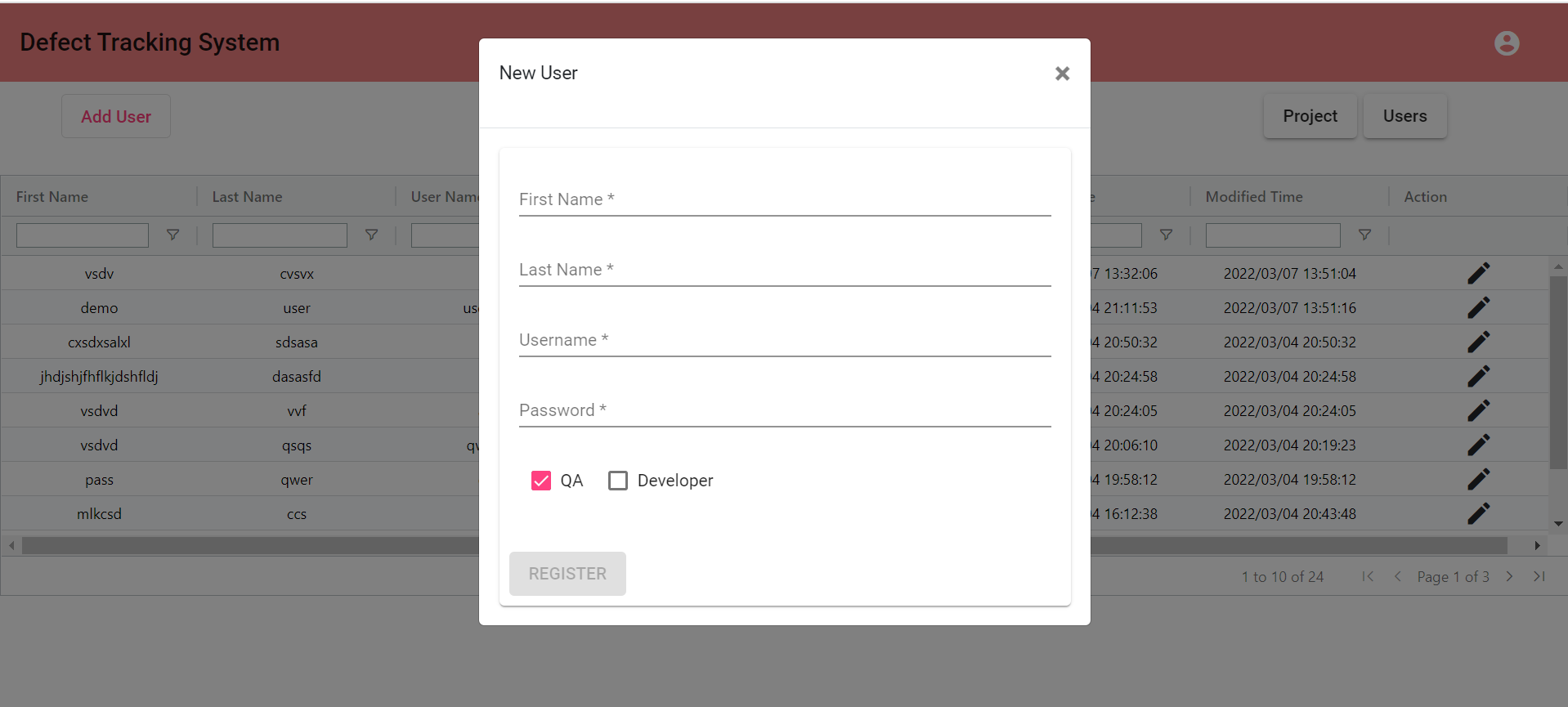
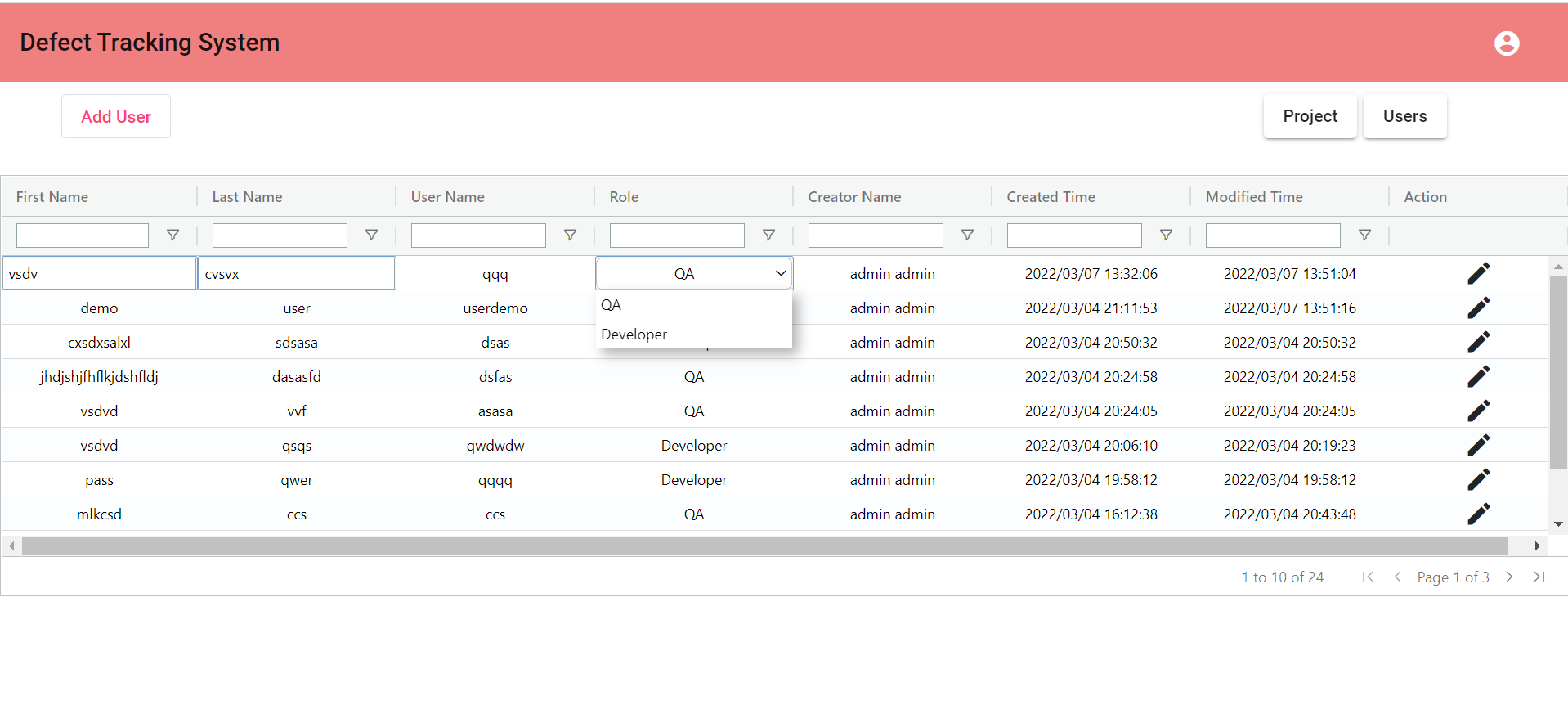
For the Authentication part, we have implemented the JWT Authentication in our project. Any user will be able to access our apis only when he will have a valid jwt token in the header. The Following diagram shows the internal working of how the JWT Token security has been added to our application.

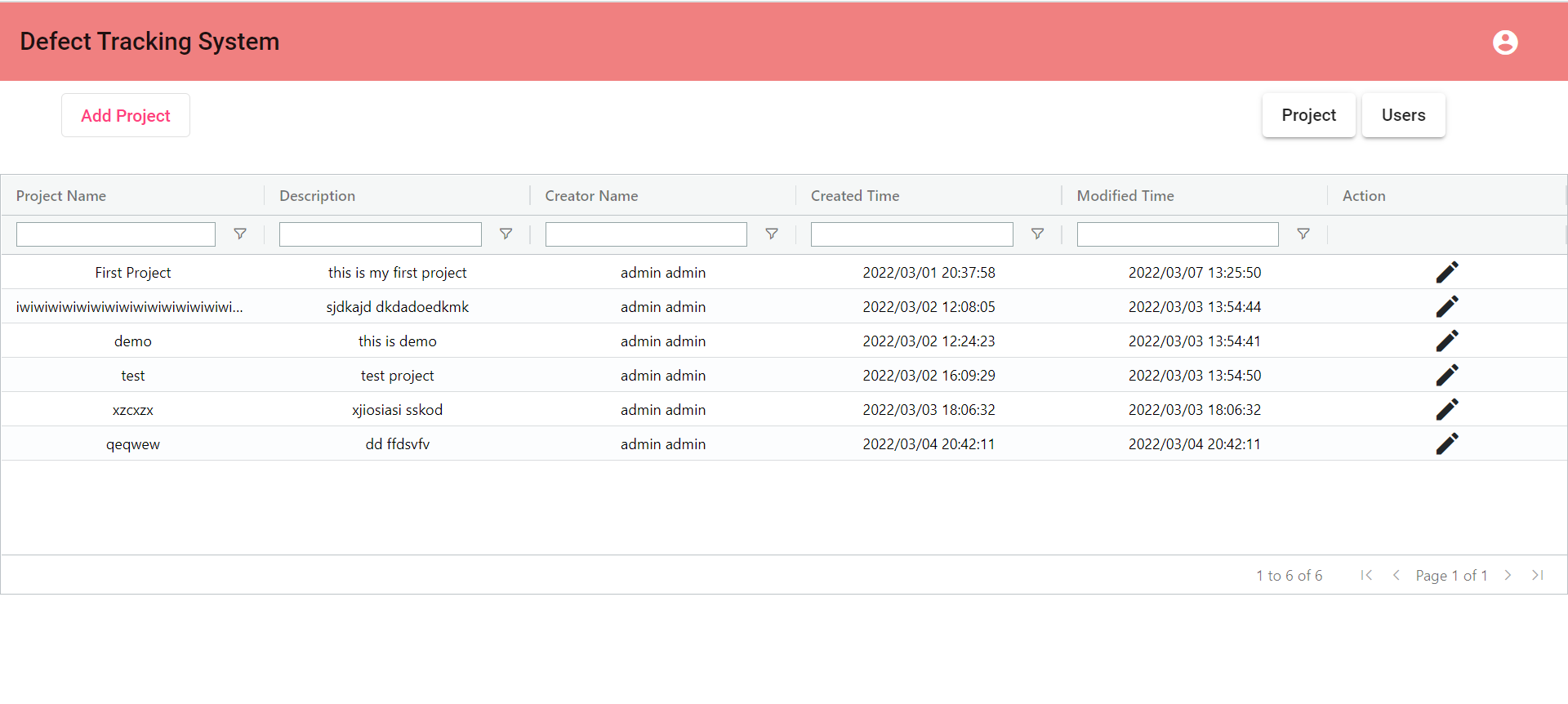
The above diagram depicts how the JWT is implemented in our DefectTracking Project. The first /login request comes in web server and it passes through the JwtRequestFilter which extends the OncePerRequestFilter. Then the request reached to LoginController which will internally call the loadUserByUsername() which will return the spring security user object along with the granted authority and then with the help of JwtTokenUtil we will generate a token out of that user object. And that JWT token will be returned back in the response.

All the further requests will have that JWT Token in the header which will be verified to give the access to that user. DefectTracking project have three User Roles –

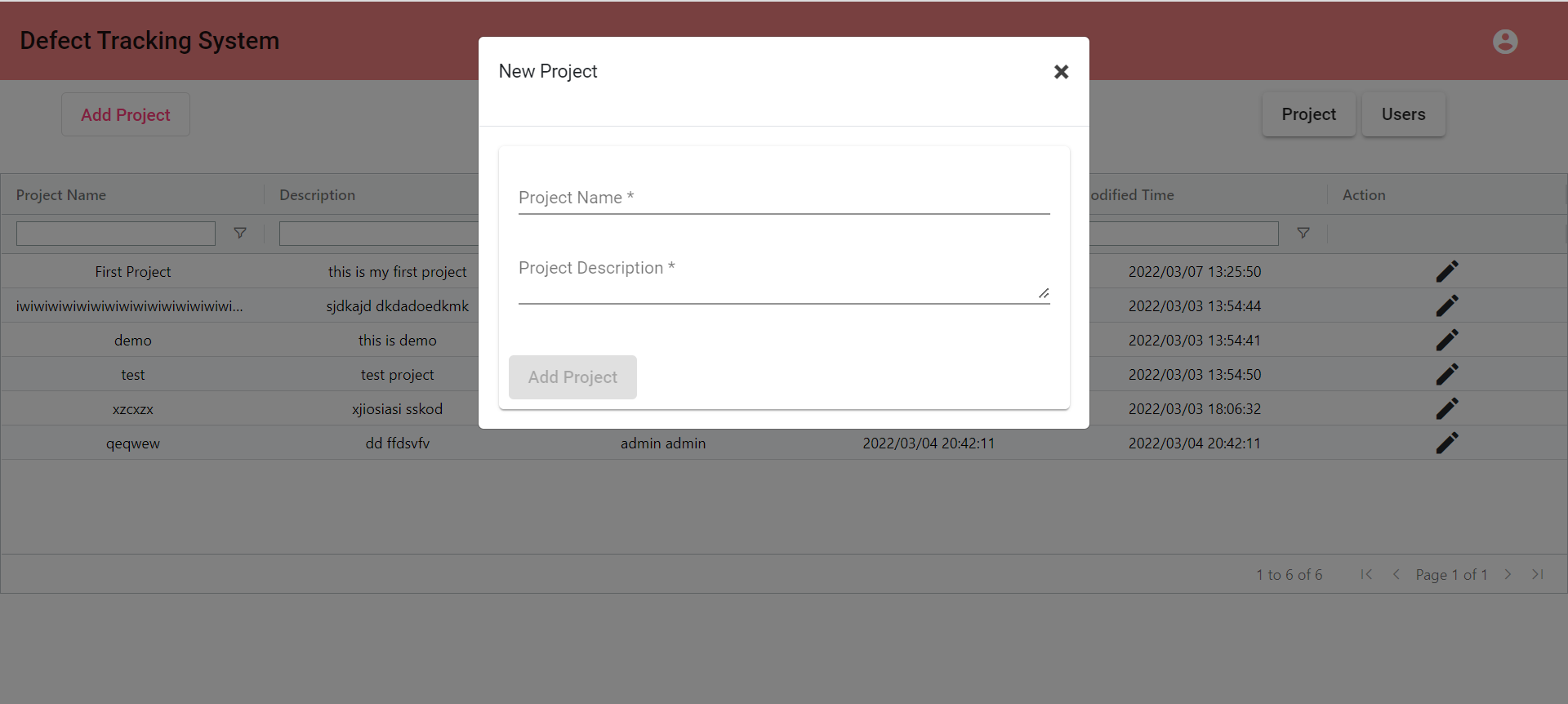
* + - ADMIN
    - QA
    - DEVELOPER

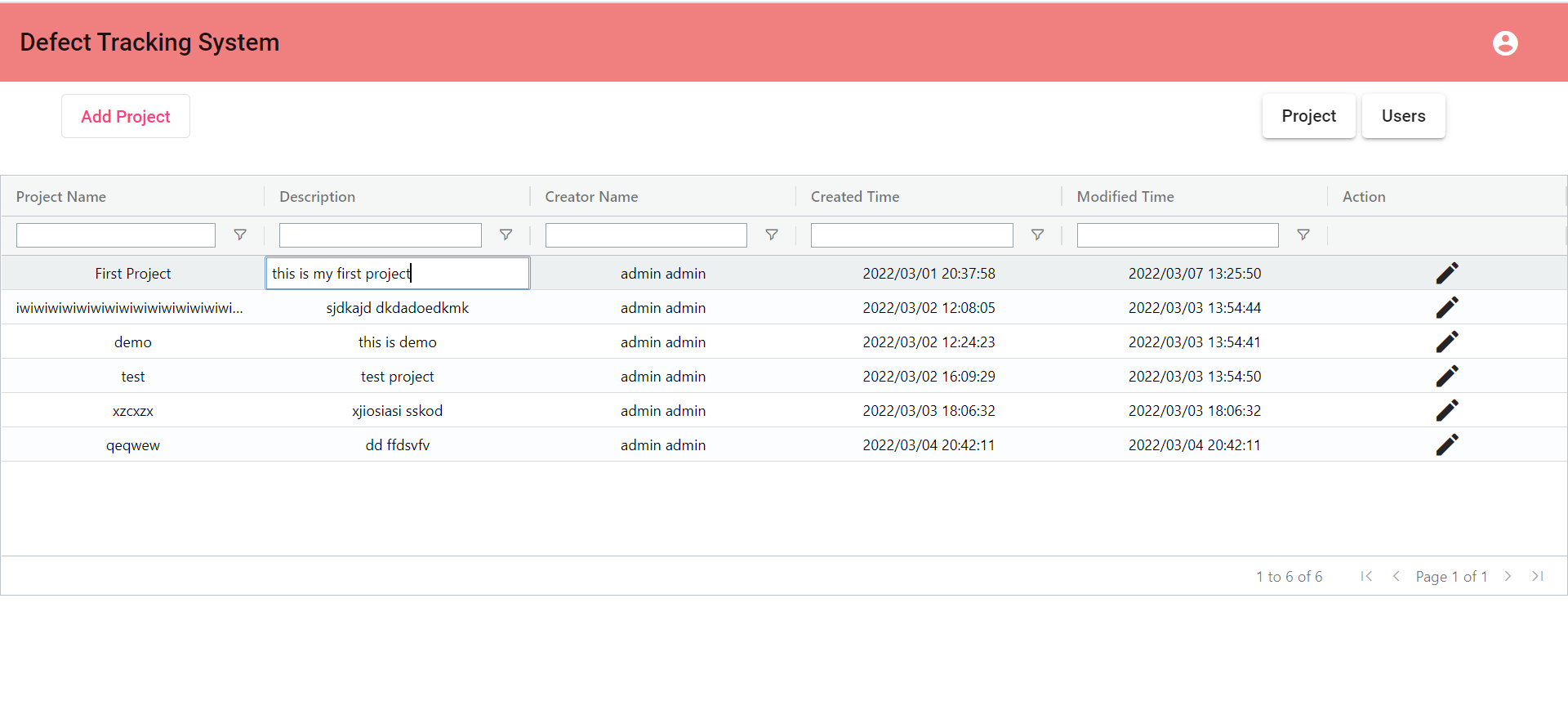
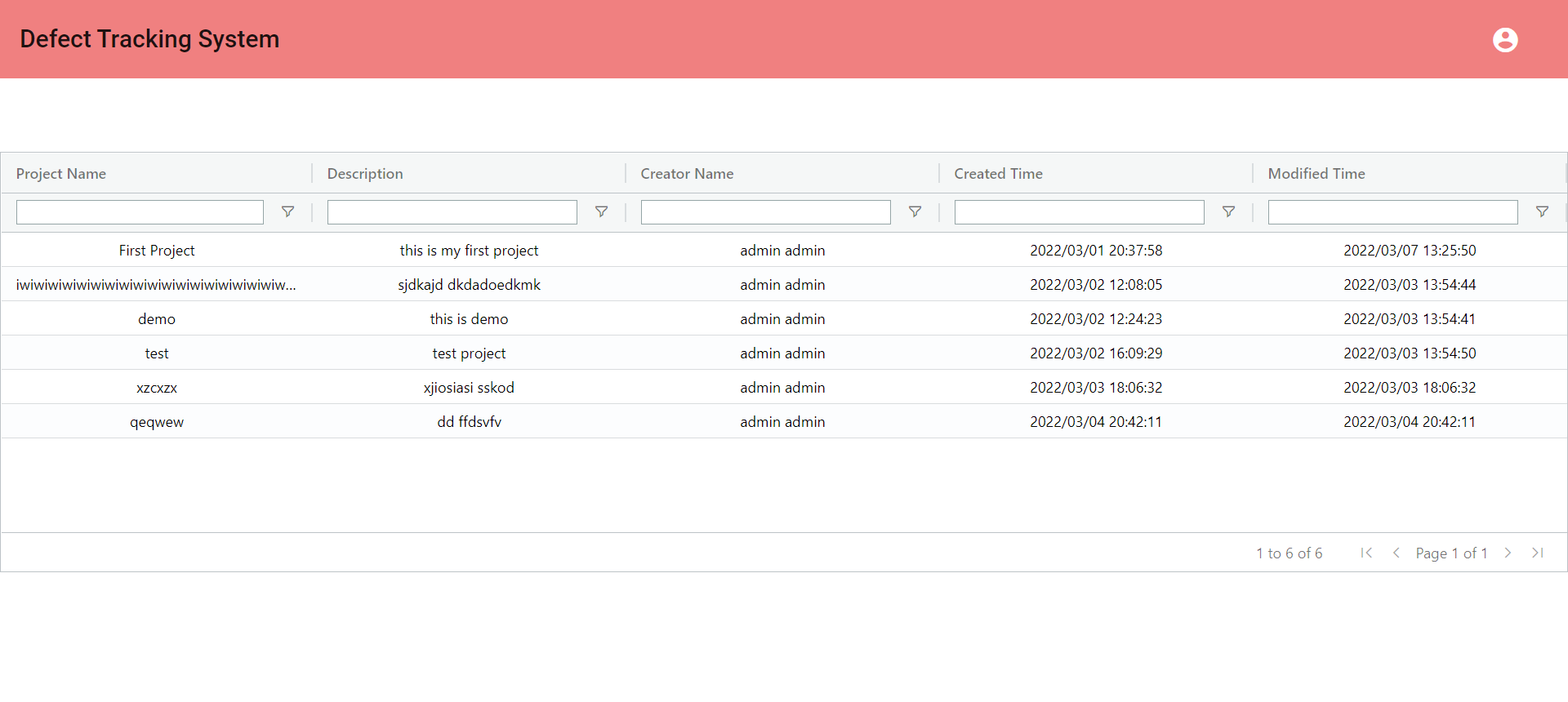
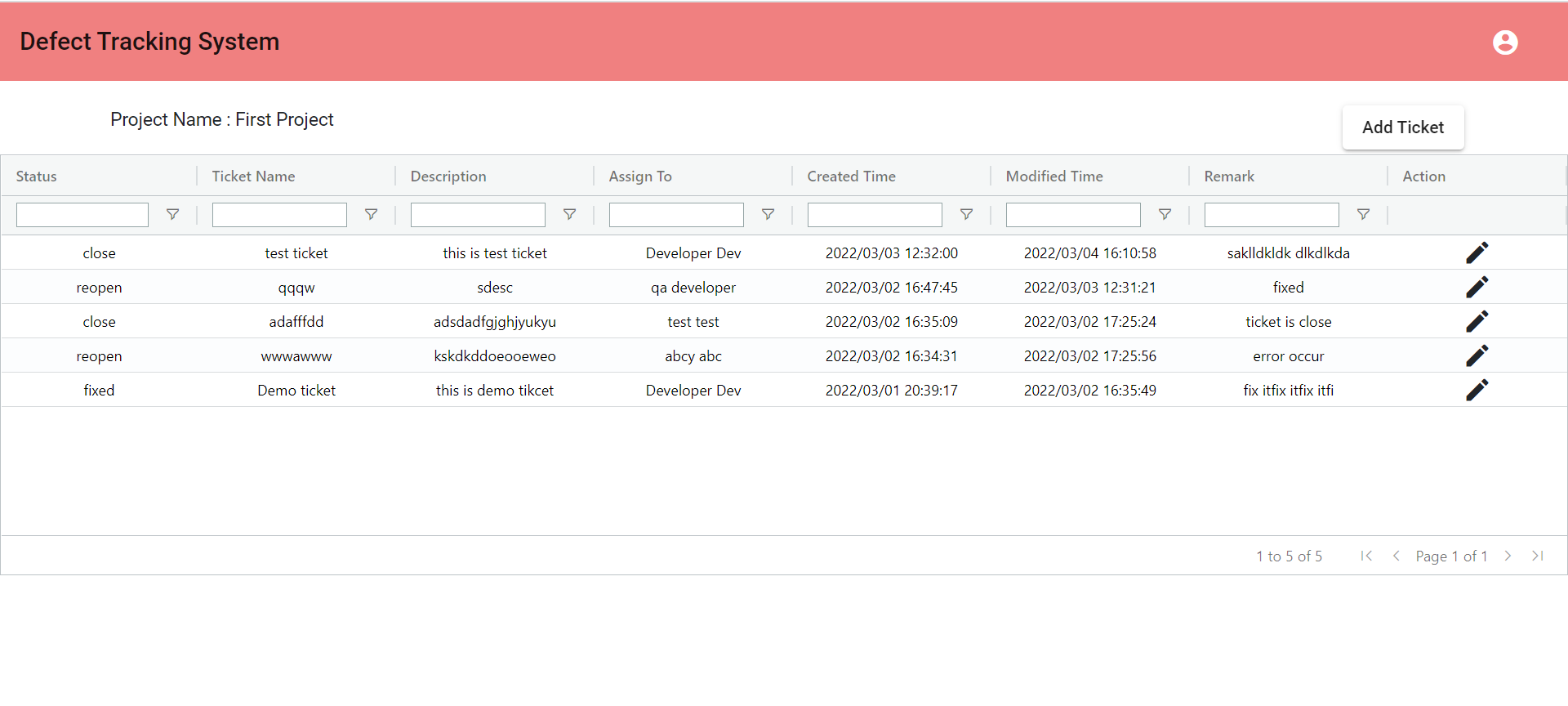
## User Interface

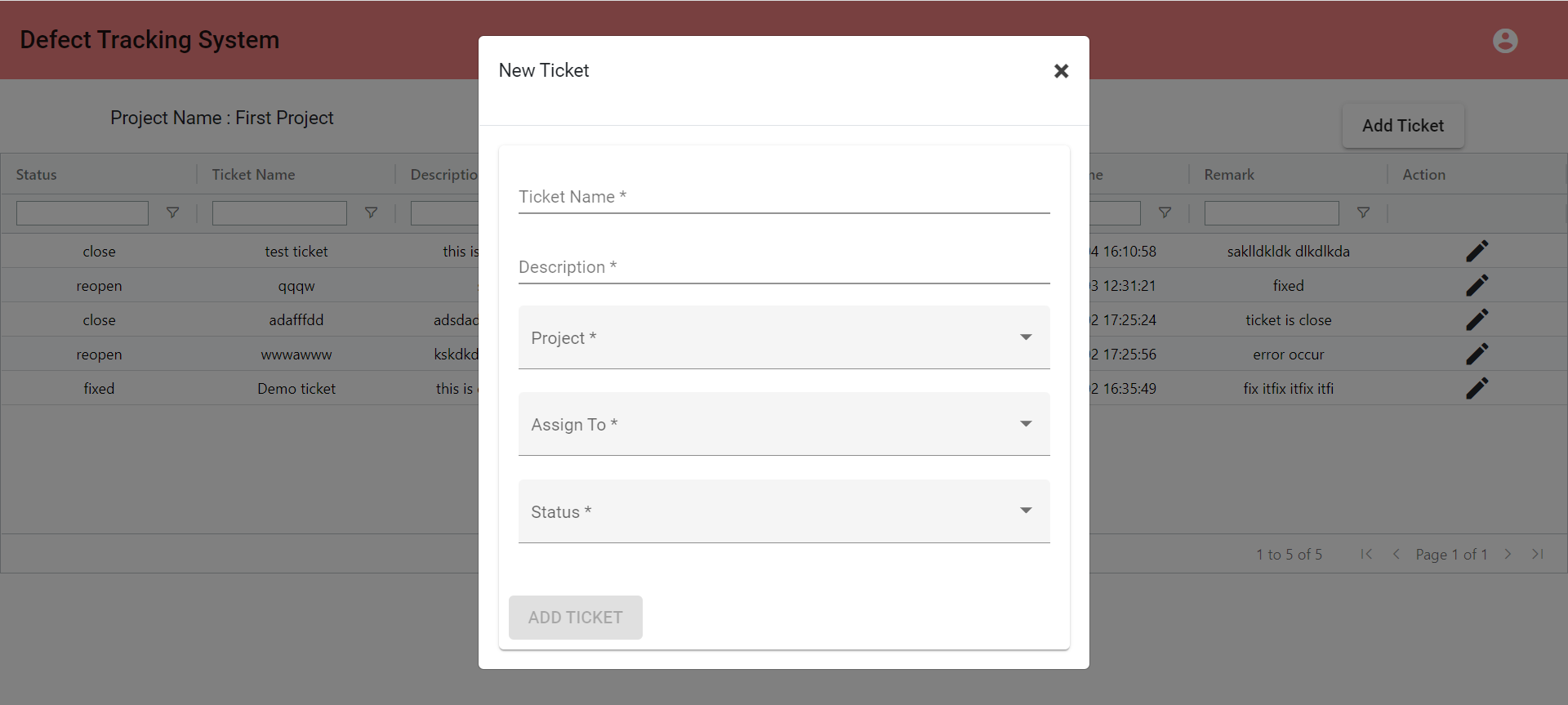
1. Login Page
2. Admin Dashboard
3. Add User page
4. User update page
5. Project Listing Page



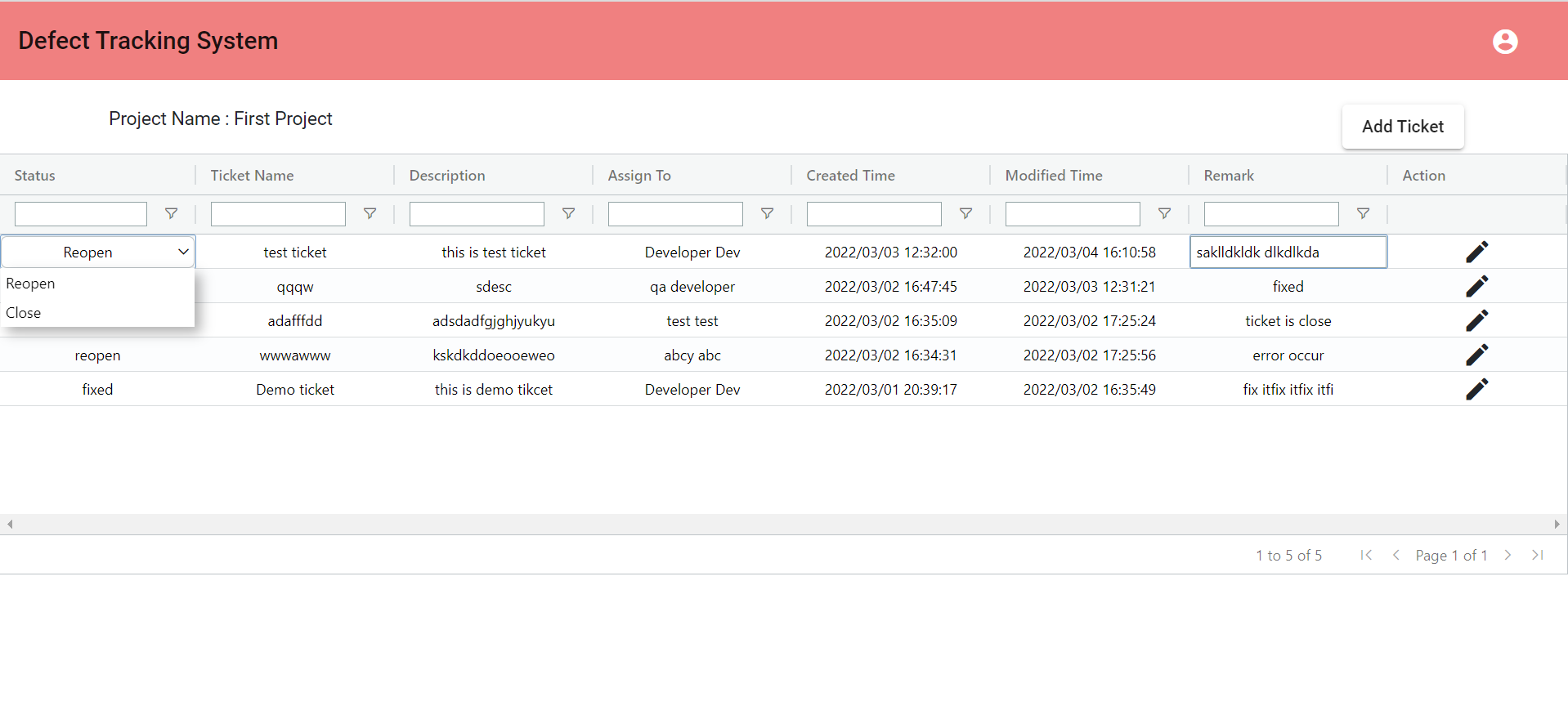
1. Project Add Page



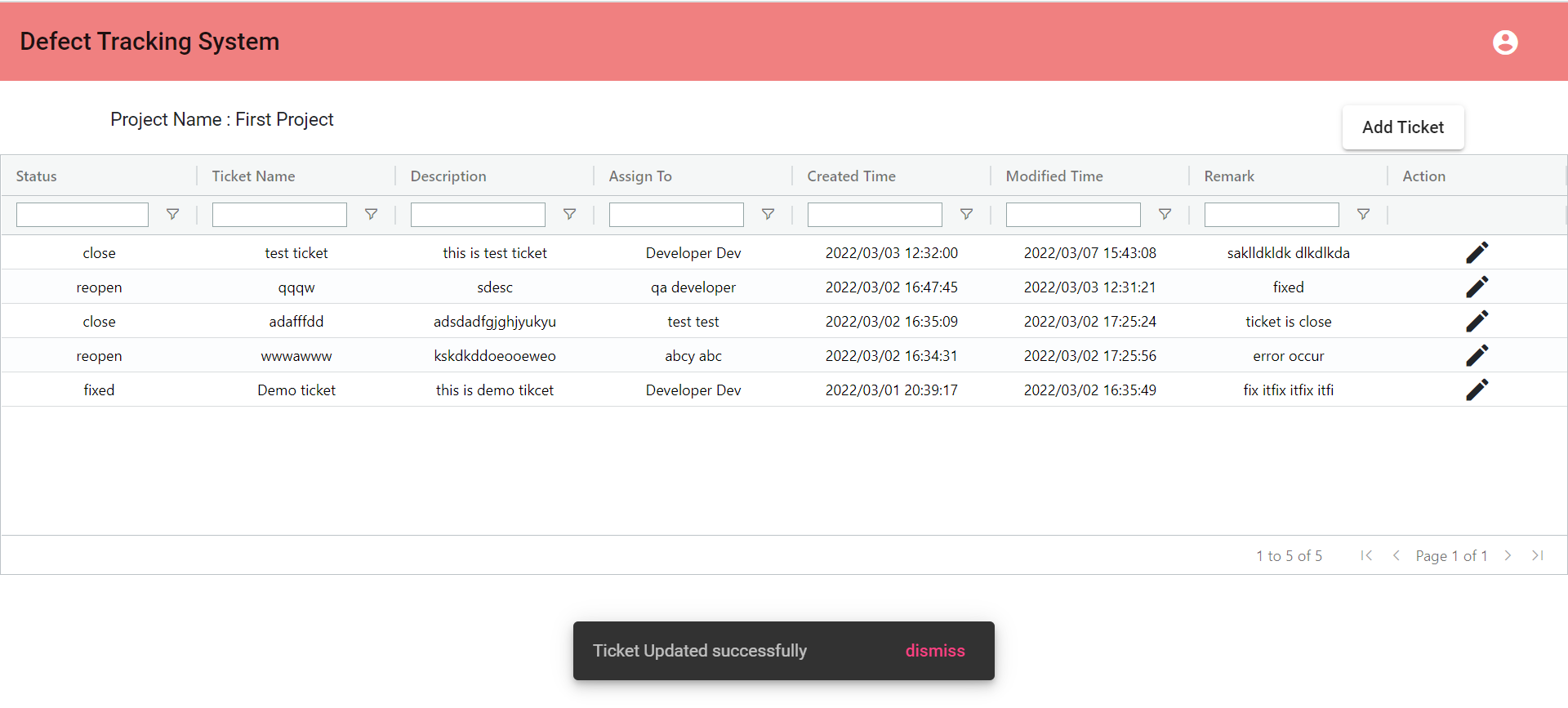
1. Project Update Page
2. QA Dashboard
3. QA Ticket Listing
4. Add Ticket Page

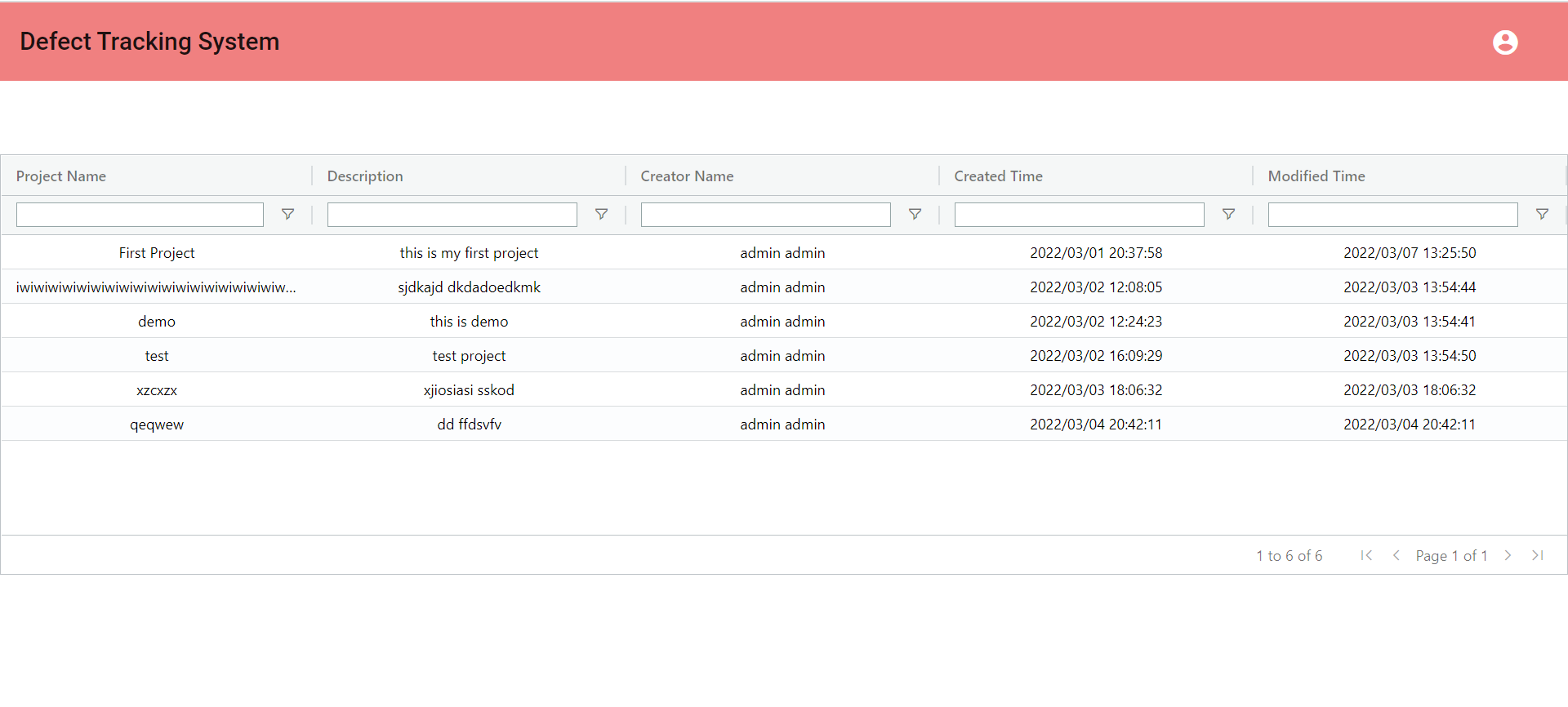
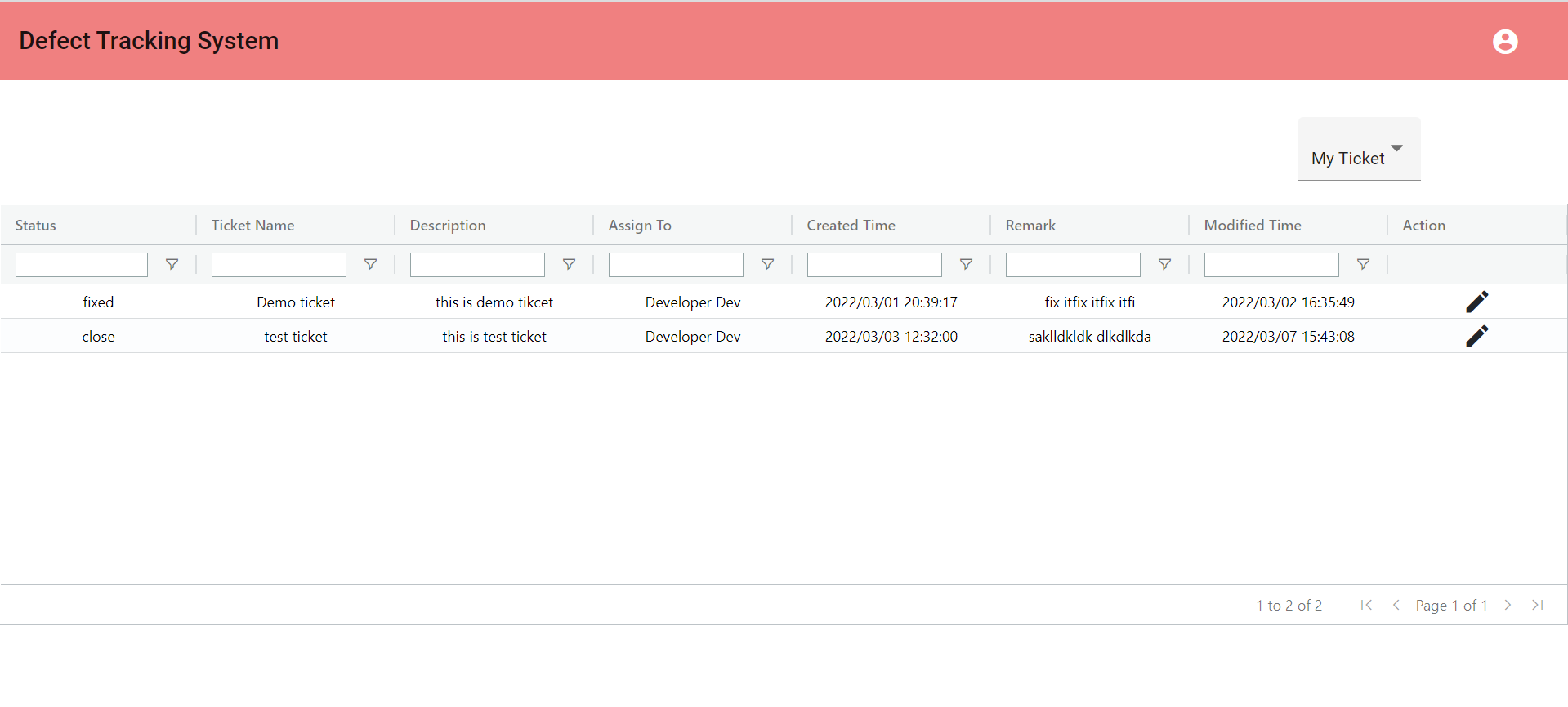
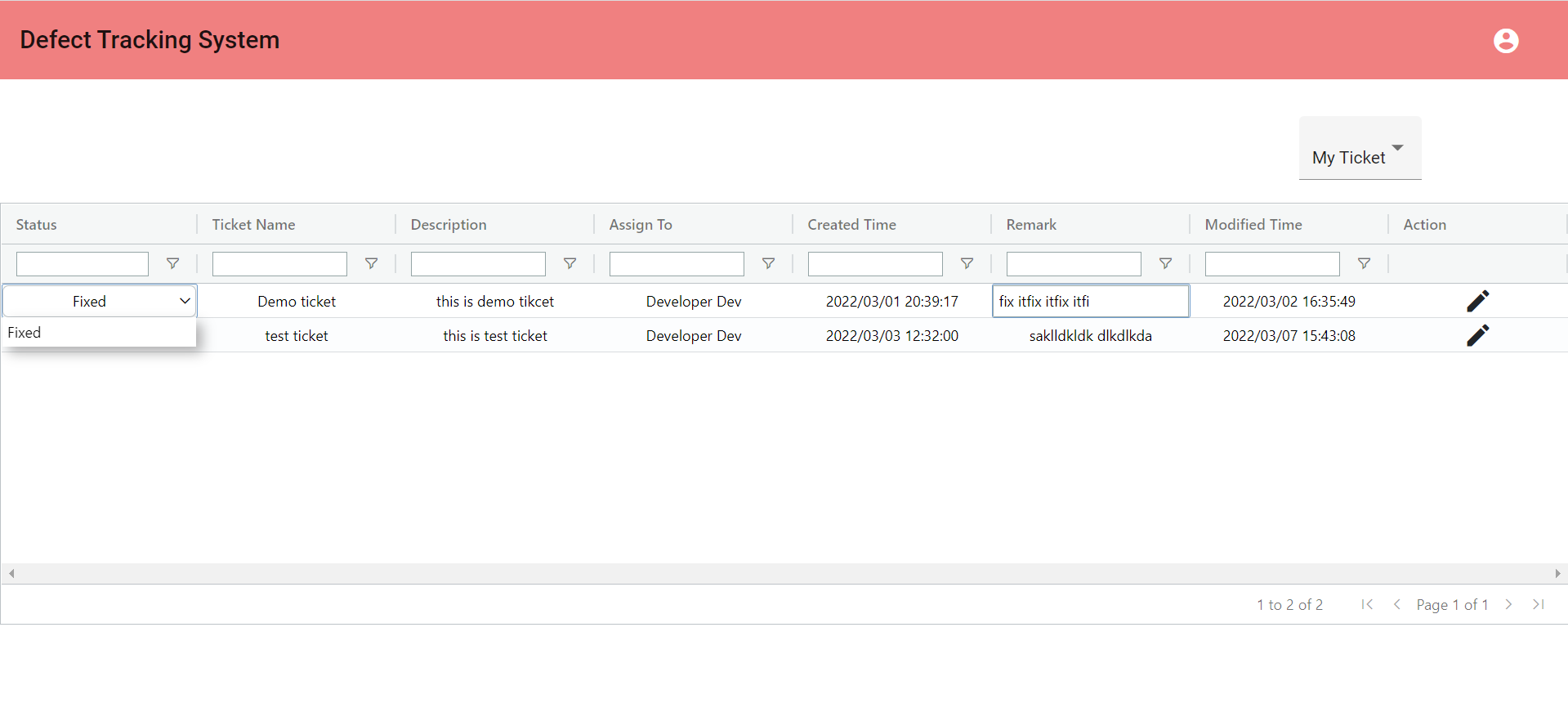
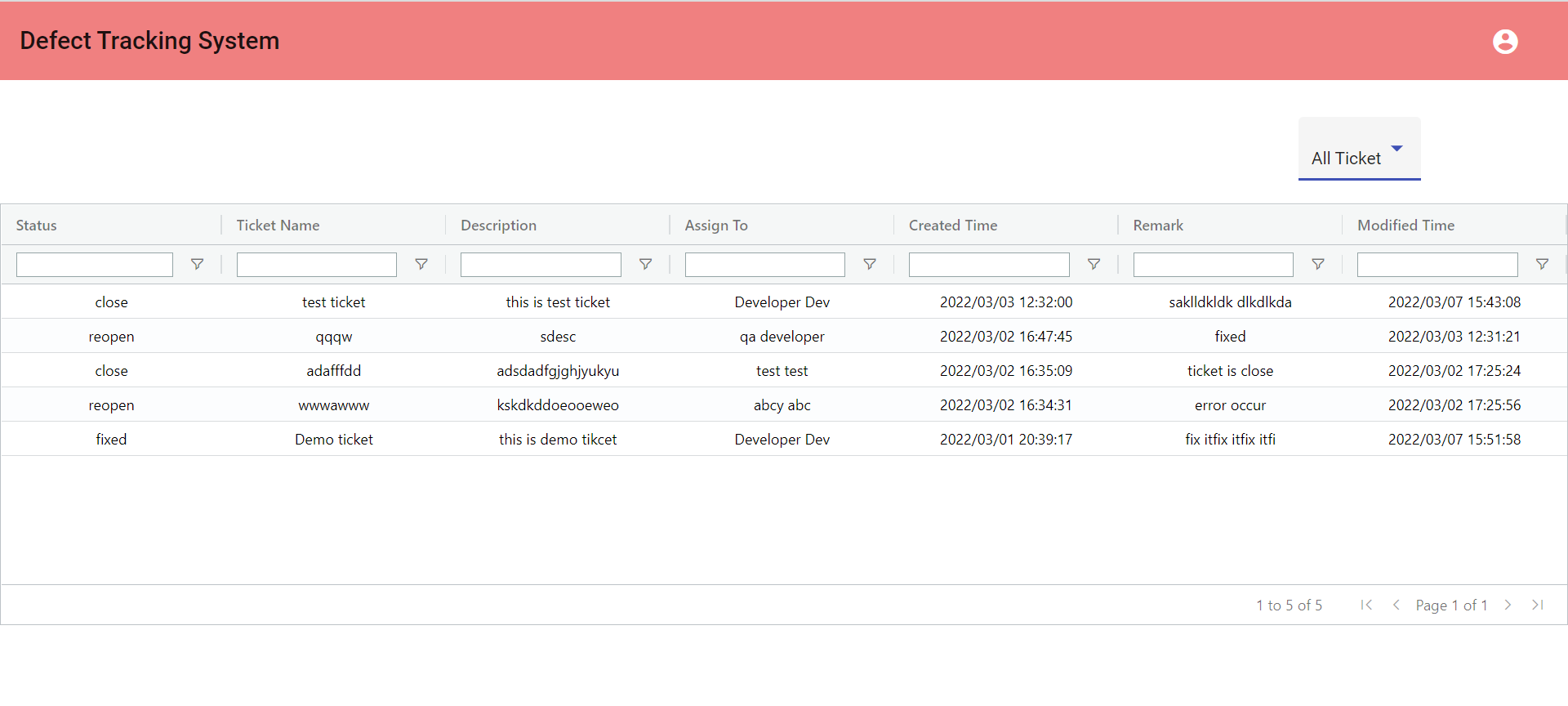


1. QA Update Ticket Page



1. Ticket Update Message Page



1. Developer Dashboard
2. Developer My Ticket Listing
3. Developer My Ticket Update Page
4. Developer All Ticket Listing

## Reference

5.1 <https://www.youtube.com/>

5.2 <https://stackoverflow.com/>

5.3 <https://github.com/>

5.4 <https://start.spring.io/>