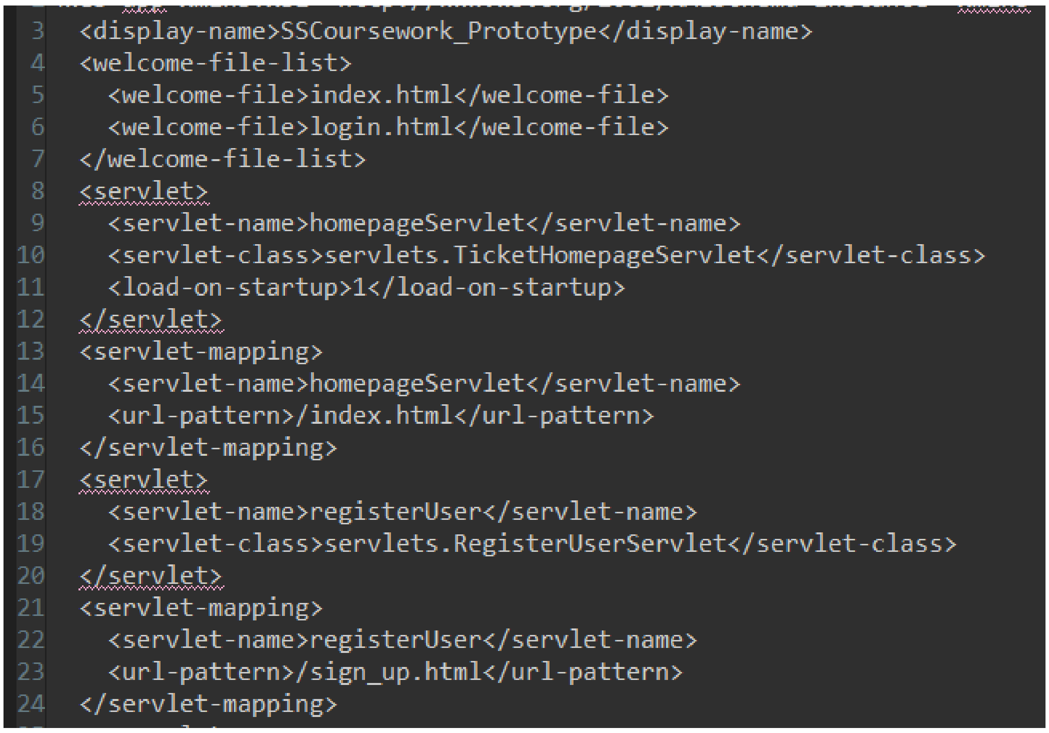
Report

Client

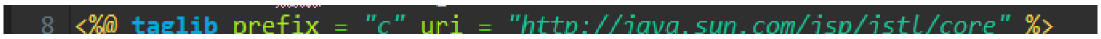
The client/web server side of the application is built using Java Servlet architecture. The user interacts with JSP pages in browser, which are generated as responses from servlets. These servlet containers are mapped using the ‘web.xml’ document.



```xml
  <display-name>SSCoursework_Prototype</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>login.html</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>homepageServlet</servlet-name>
    <servlet-class>servlets.TicketHomepageServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>homepageServlet</servlet-name>
    <url-pattern>/index.html</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>registerUser</servlet-name>
    <servlet-class>servlets.RegisterUserServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>registerUser</servlet-name>
    <url-pattern>/sign_up.html</url-pattern>
  </servlet-mapping>
```

Servlet mappings

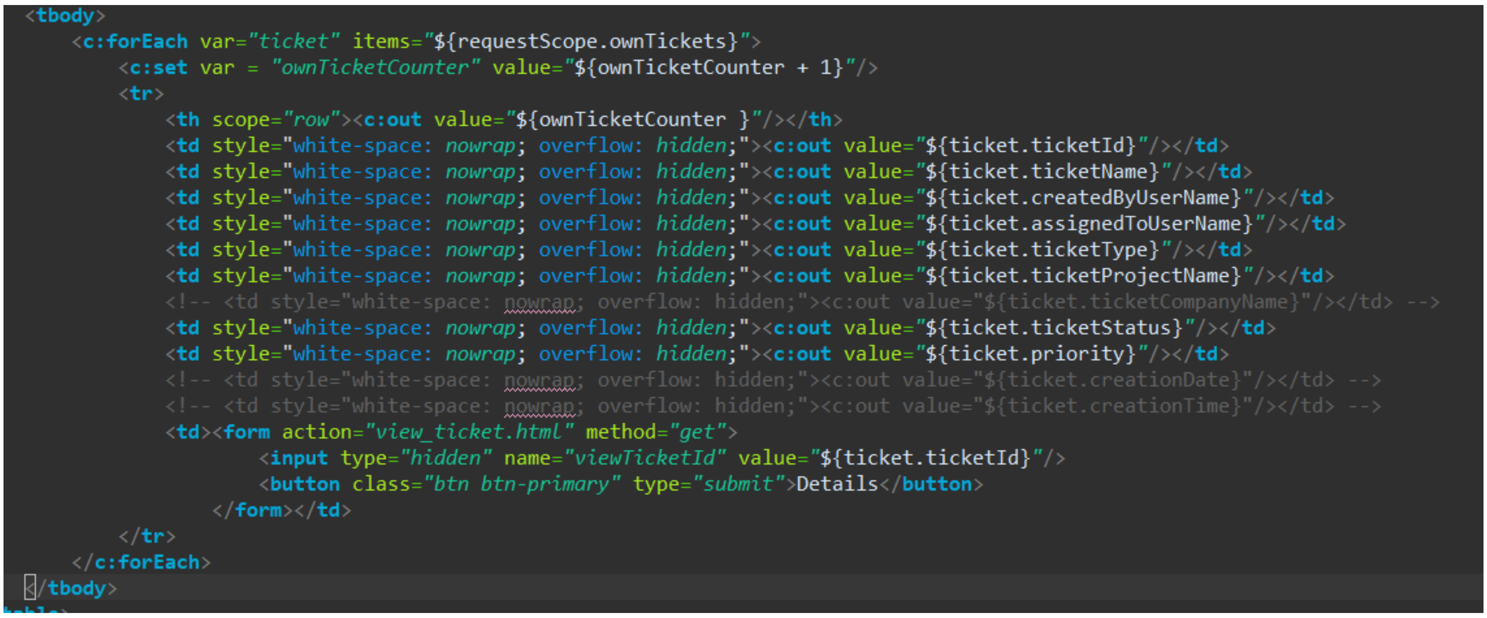
Data and business logic are processed in the web server. When this data needs to be displayed to the user, it is presented in the form of a JSTL variable.



```jsp
<%@ taglib prefix = "c" uri = "http://java.sun.com/jsp/jstl/core" %>
```

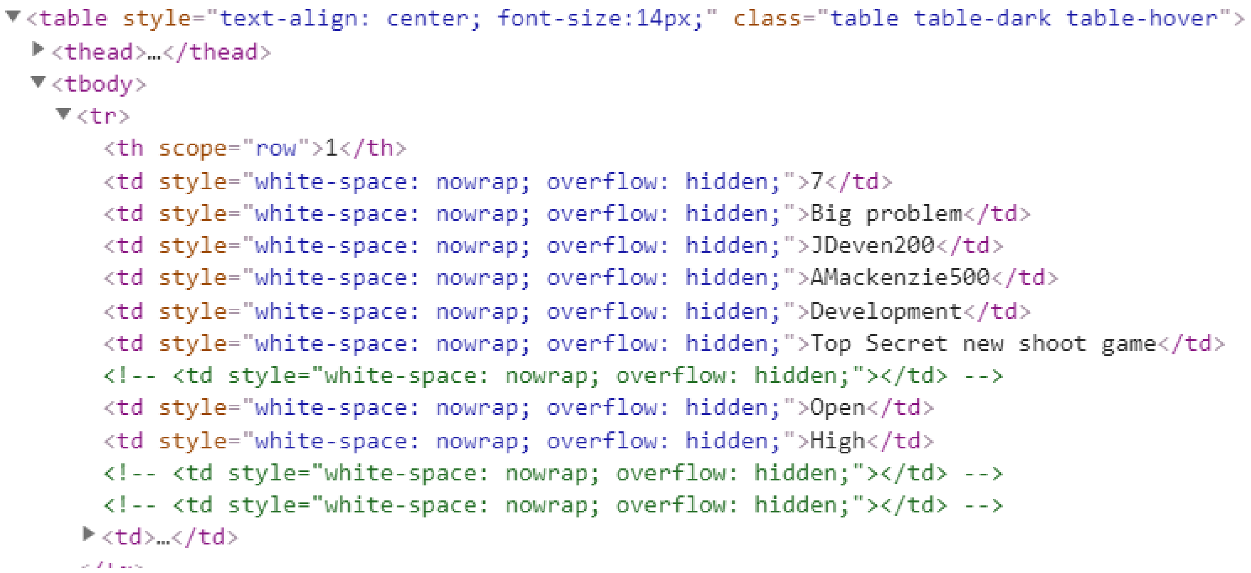
JSTL core library reference

This ensures that the presentation layer and the logic layer remain separate.



```
<tbody>
    <c:forEach var="ticket" items="${requestScope.ownTickets}">
        <c:set var = "ownTicketCounter" value="${ownTicketCounter + 1}"/>
        <tr>
            <th scope="row"><c:out value="${ownTicketCounter }"/></th>
            <td style="white-space: nowrap; overflow: hidden;"><c:out value="${ticket.ticketId}"/></td>
            <td style="white-space: nowrap; overflow: hidden;"><c:out value="${ticket.ticketName}"/></td>
            <td style="white-space: nowrap; overflow: hidden;"><c:out value="${ticket.createdByUserName}"/></td>
            <td style="white-space: nowrap; overflow: hidden;"><c:out value="${ticket.assignedToUserName}"/></td>
            <td style="white-space: nowrap; overflow: hidden;"><c:out value="${ticket.ticketType}"/></td>
            <td style="white-space: nowrap; overflow: hidden;"><c:out value="${ticket.ticketProjectName}"/></td>
            <!-- <td style="white-space: nowrap; overflow: hidden;"><c:out value="${ticket.ticketCompanyName}"/></td> -->
            <td style="white-space: nowrap; overflow: hidden;"><c:out value="${ticket.ticketStatus}"/></td>
            <td style="white-space: nowrap; overflow: hidden;"><c:out value="${ticket.priority}"/></td>
            <!-- <td style="white-space: nowrap; overflow: hidden;"><c:out value="${ticket.creationDate}"/></td> -->
            <!-- <td style="white-space: nowrap; overflow: hidden;"><c:out value="${ticket.creationTime}"/></td> -->
            <td><form action="view_ticket.html" method="get">
                    <input type="hidden" name="viewTicketId" value="${ticket.ticketId}"/>
                    <button class="btn btn-primary" type="submit">Details</button>
                </form></td>
        </tr>
    </c:forEach>
</tbody>
```

JSTL tags for a list of tickets “${ticket.ticketId}”



```
▼<table style="text-align: center; font-size:14px;" class="table table-dark table-hover">
  ▶<thead>…</thead>
  ▼<tbody>
    ▼<tr>
        <th scope="row">1</th>
        <td style="white-space: nowrap; overflow: hidden;">7</td>
        <td style="white-space: nowrap; overflow: hidden;">Big problem</td>
        <td style="white-space: nowrap; overflow: hidden;">JDeven200</td>
        <td style="white-space: nowrap; overflow: hidden;">AMackenzie500</td>
        <td style="white-space: nowrap; overflow: hidden;">Development</td>
        <td style="white-space: nowrap; overflow: hidden;">Top Secret new shoot game</td>
        <!-- <td style="white-space: nowrap; overflow: hidden;"></td> -->
        <td style="white-space: nowrap; overflow: hidden;">Open</td>
        <td style="white-space: nowrap; overflow: hidden;">High</td>
        <!-- <td style="white-space: nowrap; overflow: hidden;"></td> -->
        <!-- <td style="white-space: nowrap; overflow: hidden;"></td> -->
      ▶<td>…</td>
```

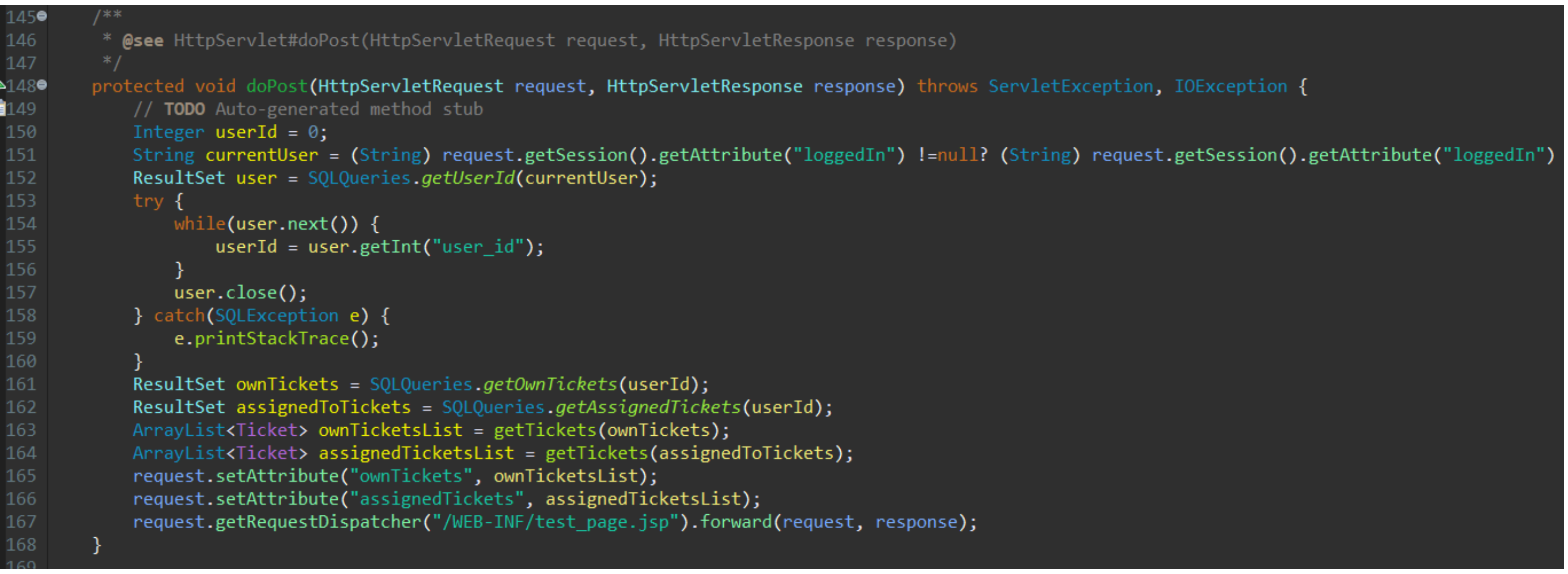
HTML of JSP shown to client

Server

The database tables are split into “Users”, “User Credentials”, “Access Rights”, “Roles”, “Company”, “Project”, “Ticket”, “Ticket Comments” and “Ticket Log File”. For this prototype, the “Ticket Comments” and “Ticket Log File” did not get used. These tables are accessed using a Postgres JDBC. The reason for separating the tables is because if a single table is broken into, the other tables may remain secure.

Data

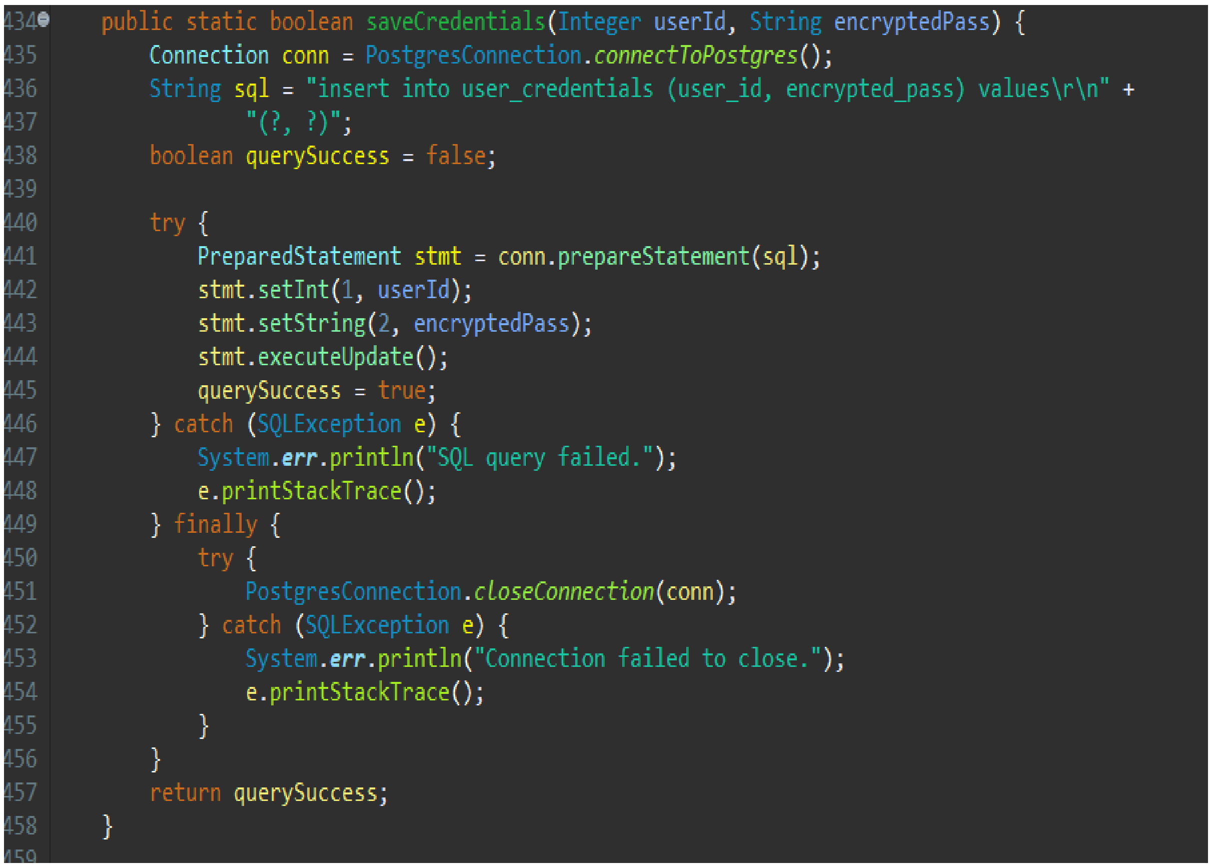
The logic layer which receives user requests, processes them and generates a response page is all handled using Java. When a request is received by a servlet, user inputs are parameterised (if any exist), the appropriate methods are called, the database server is accessed, the relevant SQL queries are called, and a response page is generated.



```java
    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        Integer userId = 0;
        String currentUser = (String) request.getSession().getAttribute("loggedIn") !=null? (String) request.getSession().getAttribute("loggedIn")
        ResultSet user = SQLQueries.getUserId(currentUser);
        try {
            while(user.next()) {
                userId = user.getInt("user_id");
            }
            user.close();
        } catch(SQLException e) {
            e.printStackTrace();
        }
        ResultSet ownTickets = SQLQueries.getOwnTickets(userId);
        ResultSet assignedToTickets = SQLQueries.getAssignedTickets(userId);
        ArrayList<Ticket> ownTicketsList = getTickets(ownTickets);
        ArrayList<Ticket> assignedTicketsList = getTickets(assignedToTickets);
        request.setAttribute("ownTickets", ownTicketsList);
        request.setAttribute("assignedTickets", assignedTicketsList);
        request.getRequestDispatcher("/WEB-INF/test_page.jsp").forward(request, response);
    }
```

Example of a POST request handler in the 'view_tickets.html' servlet

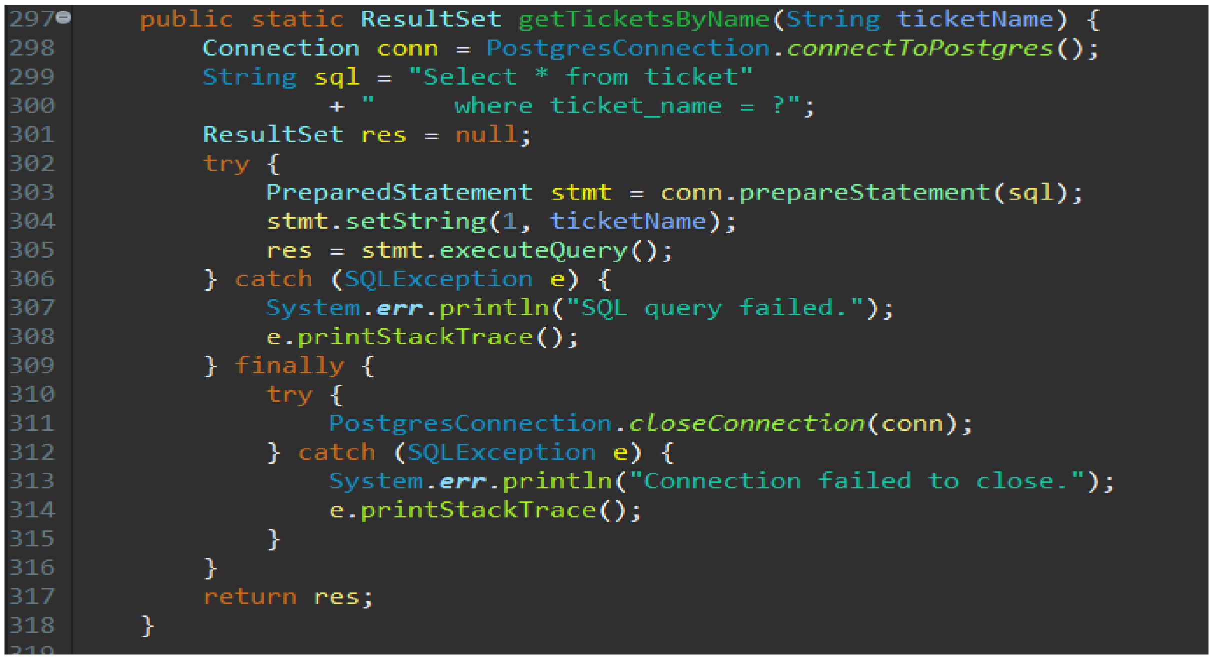
The SQL queries are hard coded into a Java class, each query is contained within its own method. An insert query returns a Boolean to check if it has worked or not and a select query is returned as a ResultSet object.



```java
    public static boolean saveCredentials(Integer userId, String encryptedPass) {
        Connection conn = PostgresConnection.connectToPostgres();
        String sql = "insert into user_credentials (user_id, encrypted_pass) values\r\n" +
                "(?, ?)";
        boolean querySuccess = false;

        try {
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setInt(1, userId);
            stmt.setString(2, encryptedPass);
            stmt.executeUpdate();
            querySuccess = true;
        } catch (SQLException e) {
            System.err.println("SQL query failed.");
            e.printStackTrace();
        } finally {
            try {
                PostgresConnection.closeConnection(conn);
            } catch (SQLException e) {
                System.err.println("Connection failed to close.");
                e.printStackTrace();
            }
        }
        return querySuccess;
    }
```

Example of an insert query method



```java
    public static ResultSet getTicketsByName(String ticketName) {
        Connection conn = PostgresConnection.connectToPostgres();
        String sql = "Select * from ticket"
                + "     where ticket_name = ?";
        ResultSet res = null;
        try {
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString(1, ticketName);
            res = stmt.executeQuery();
        } catch (SQLException e) {
            System.err.println("SQL query failed.");
            e.printStackTrace();
        } finally {
            try {
                PostgresConnection.closeConnection(conn);
            } catch (SQLException e) {
                System.err.println("Connection failed to close.");
                e.printStackTrace();
            }
        }
        return res;
    }
```

Example of a select query method

Security Controls

Authentication

A session attribute and a user cookie are scanned for using an authorisation filter whenever a page from the application is requested. These are provided upon successful login.

Passwords are stored in the “User Credentials” table and are associated with users based on an ID number.

To mitigate weak passwords, at least one letter and number must be present as well as one special character. A password can only be between 8 to 16 characters long.

Authorisation

Every time ticket details are requested, or a ticket is created, the user who made the request is checked. Their role is then found along with the privileges that they have. If the action they attempted does not fall in line with their privileges, the action will not occur.

Repudiation

A log of every request along with the user who made the request is made. For the prototype, this is contained within the console, but for a full application it would be saved to a text file.

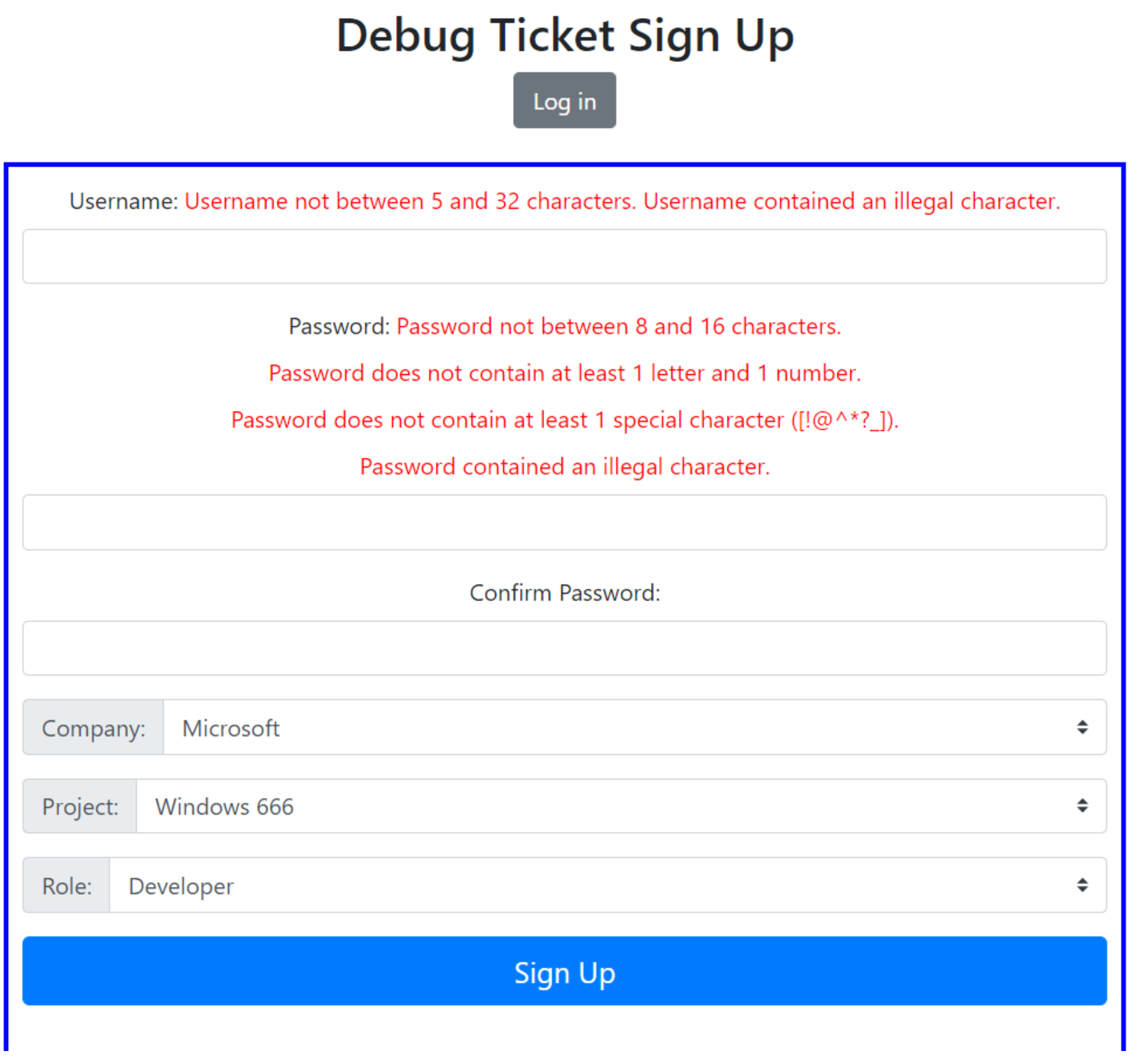
SQL Injection/XSS Mitigation

Every input a user makes is parameterised through prepared statements. As a bonus, a list of illegal characters (e.g. <>`’*) are used to sanitise data sent/received to and from the database. This sanitisation process is also used to mitigate XSS attacks.

Word Count: 498

Screenshots

Sign Up



Debug Ticket Sign Up

Username: Username not between 5 and 32 characters. Username contained an illegal character.

Password: Password not between 8 and 16 characters.

Password does not contain at least 1 letter and 1 number.

Password does not contain at least 1 special character ([!@^*?_]).

Password contained an illegal character.

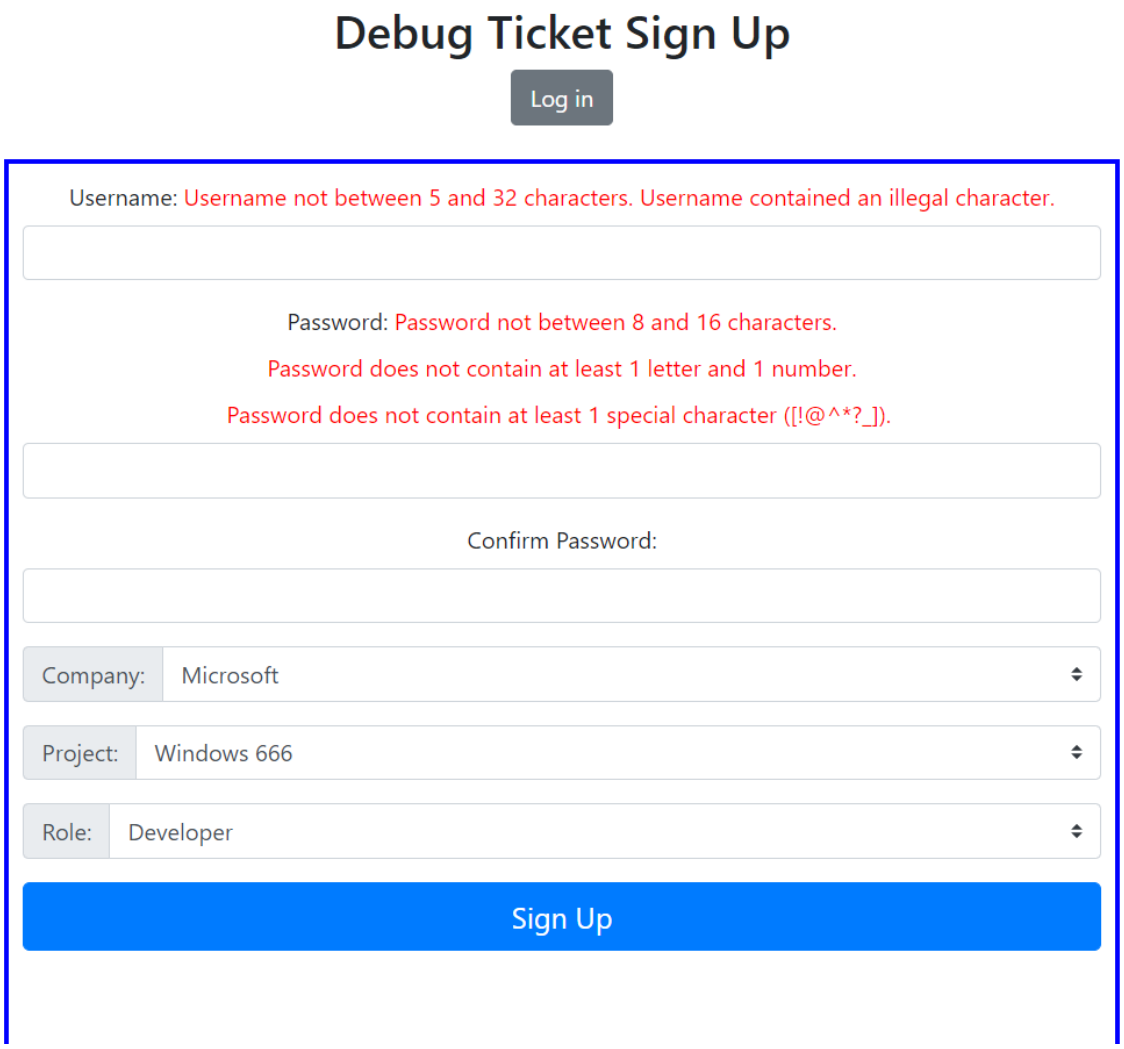
Confirm Password:

Company: Microsoft

Project: Windows 666

Role: Developer

username/password length incorrect, password not alphanumeric, password has no special character, username and password contained illegal character



Debug Ticket Sign Up

Log in

Username: Username not between 5 and 32 characters. Username contained an illegal character.

Password: Password not between 8 and 16 characters.

Password does not contain at least 1 letter and 1 number.

Password does not contain at least 1 special character ([!@^*?_]).

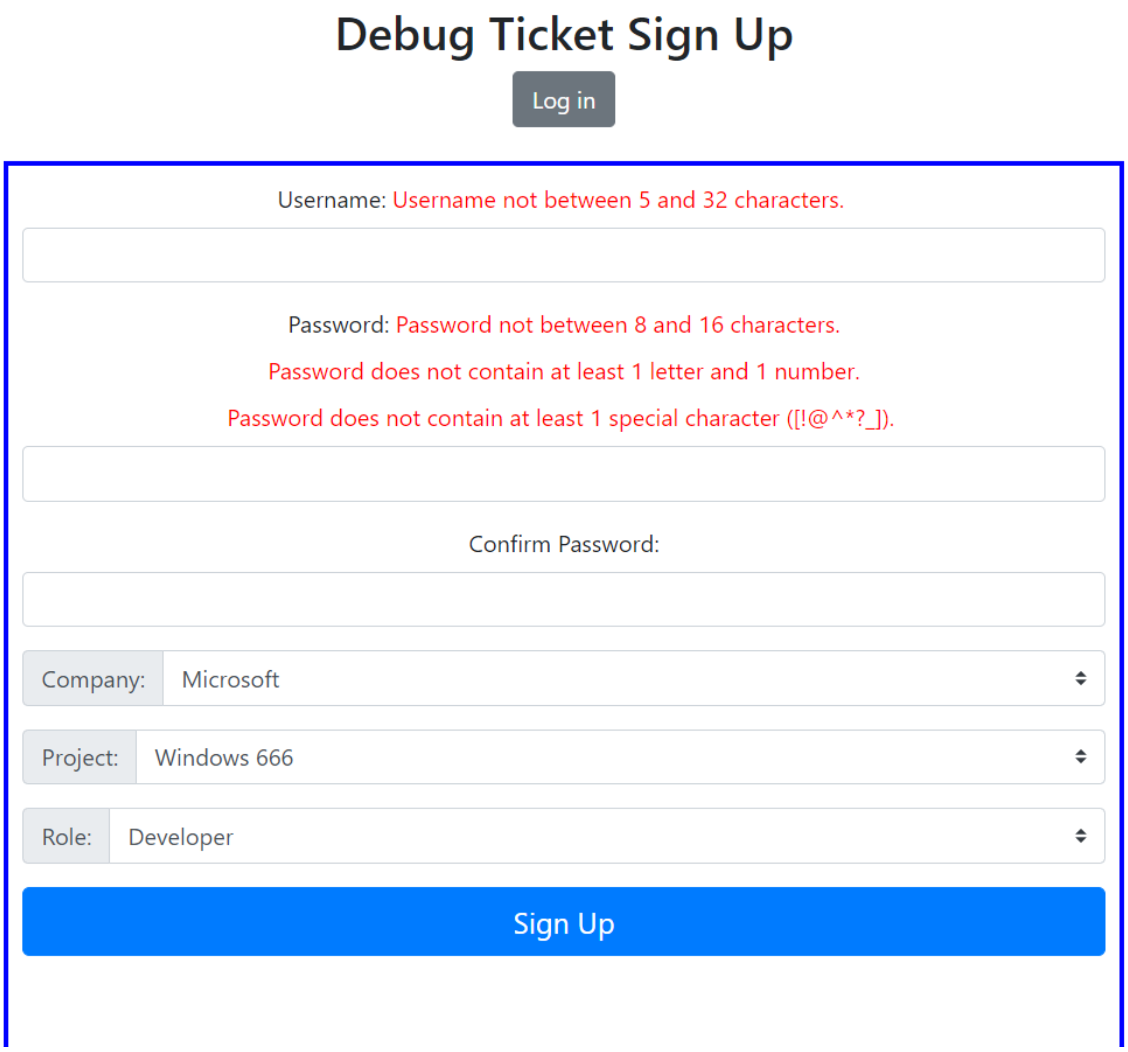
Confirm Password:

Company: Microsoft

Project: Windows 666

Role: Developer

username/password length incorrect, password not alphanumeric, password has no special character, username contained illegal character


Debug Ticket Sign Up

Username: Username not between 5 and 32 characters.

Password: Password not between 8 and 16 characters.

Password does not contain at least 1 letter and 1 number.

Password does not contain at least 1 special character ([!@^*?_]).

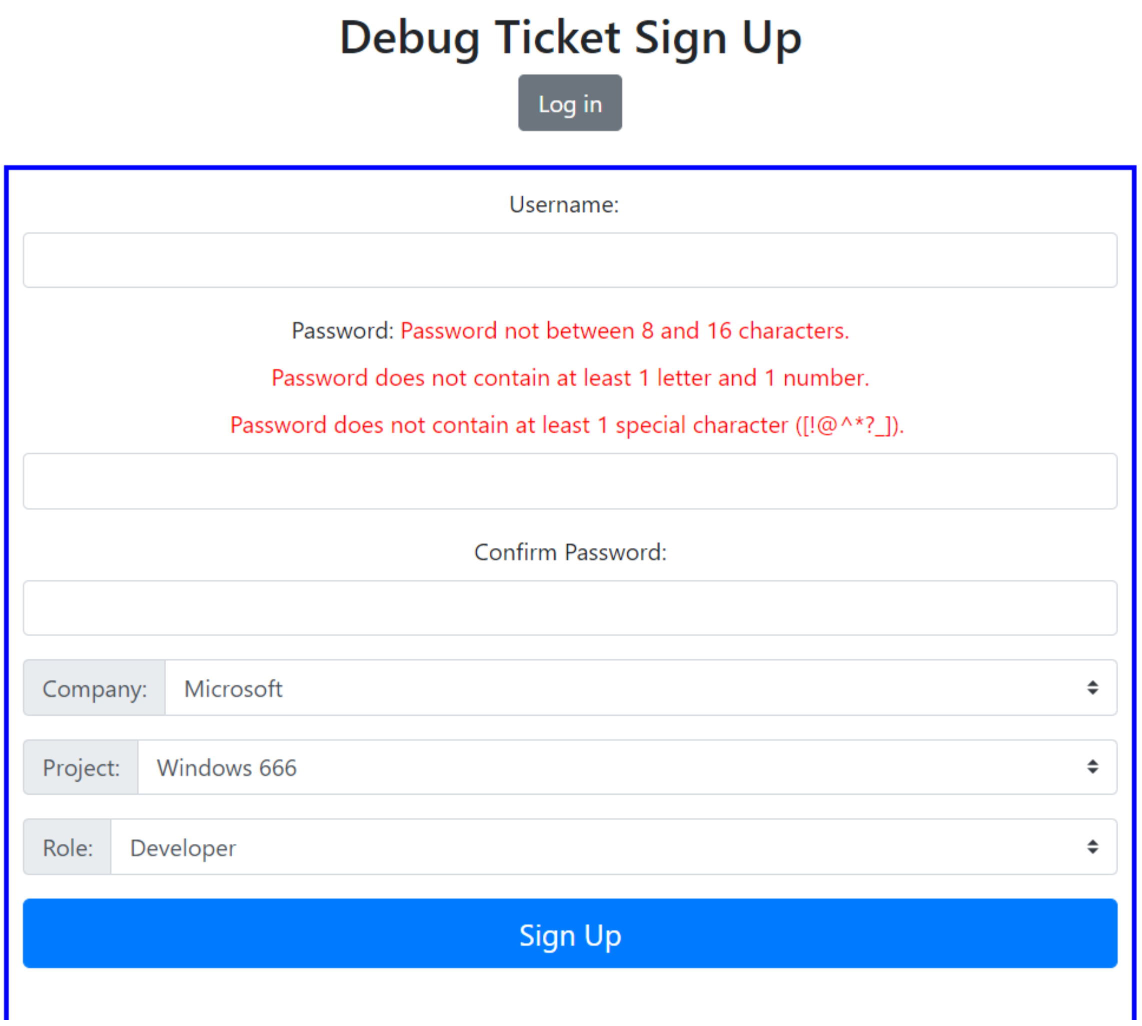
Confirm Password:

Company: Microsoft

Project: Windows 666

Role: Developer

username/password length incorrect, password not alphanumeric, password has no special character



Debug Ticket Sign Up

Username:

Password: Password not between 8 and 16 characters.

Password does not contain at least 1 letter and 1 number.

Password does not contain at least 1 special character ([!@^*?_]).

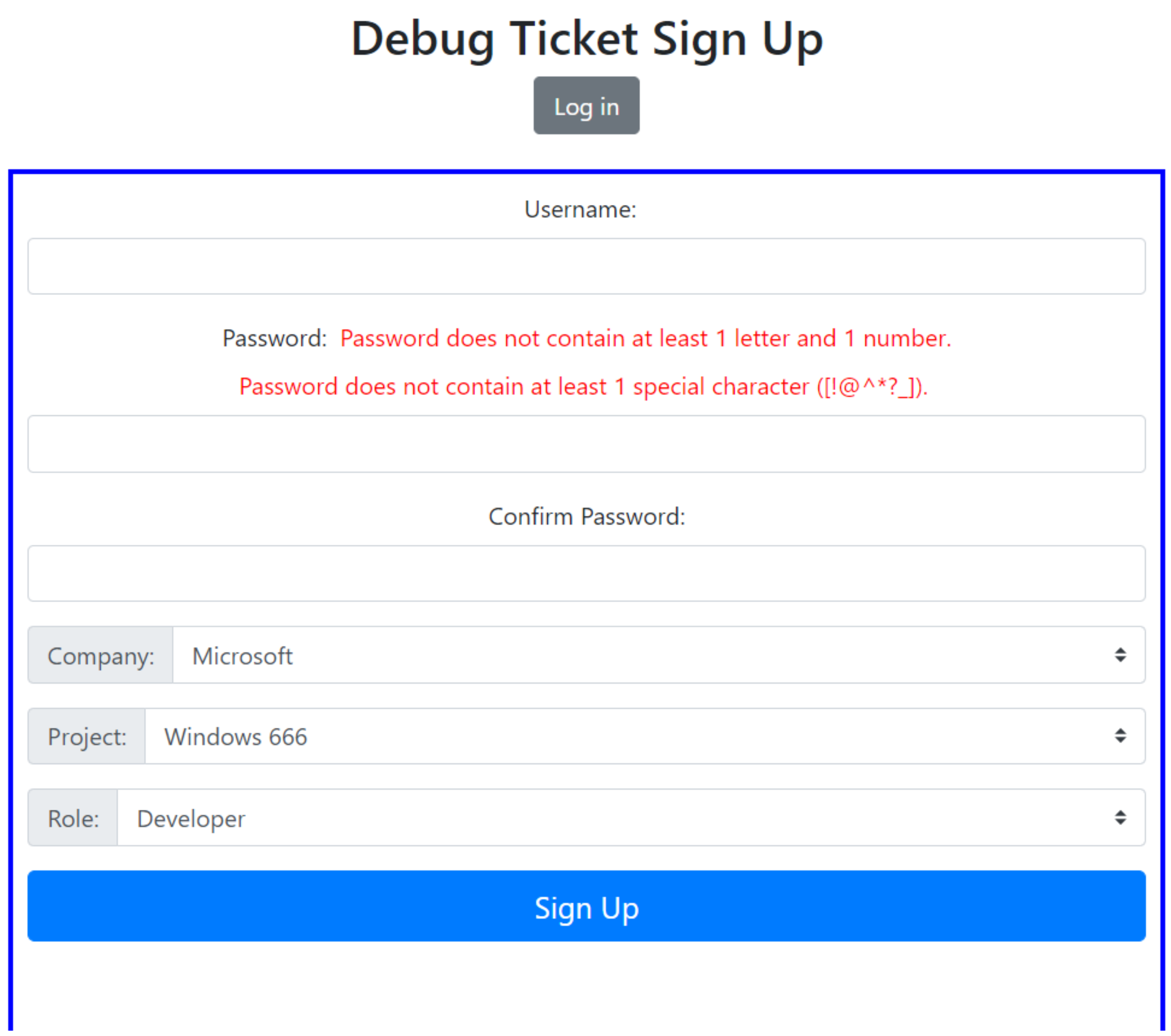
Confirm Password:

Company: Microsoft

Project: Windows 666

Role: Developer

password length incorrect, password not alphanumeric, password has no special character


Debug Ticket Sign Up

Log in

Username:

Password: Password does not contain at least 1 letter and 1 number.

Password does not contain at least 1 special character ([!@^*?_]).

Confirm Password:

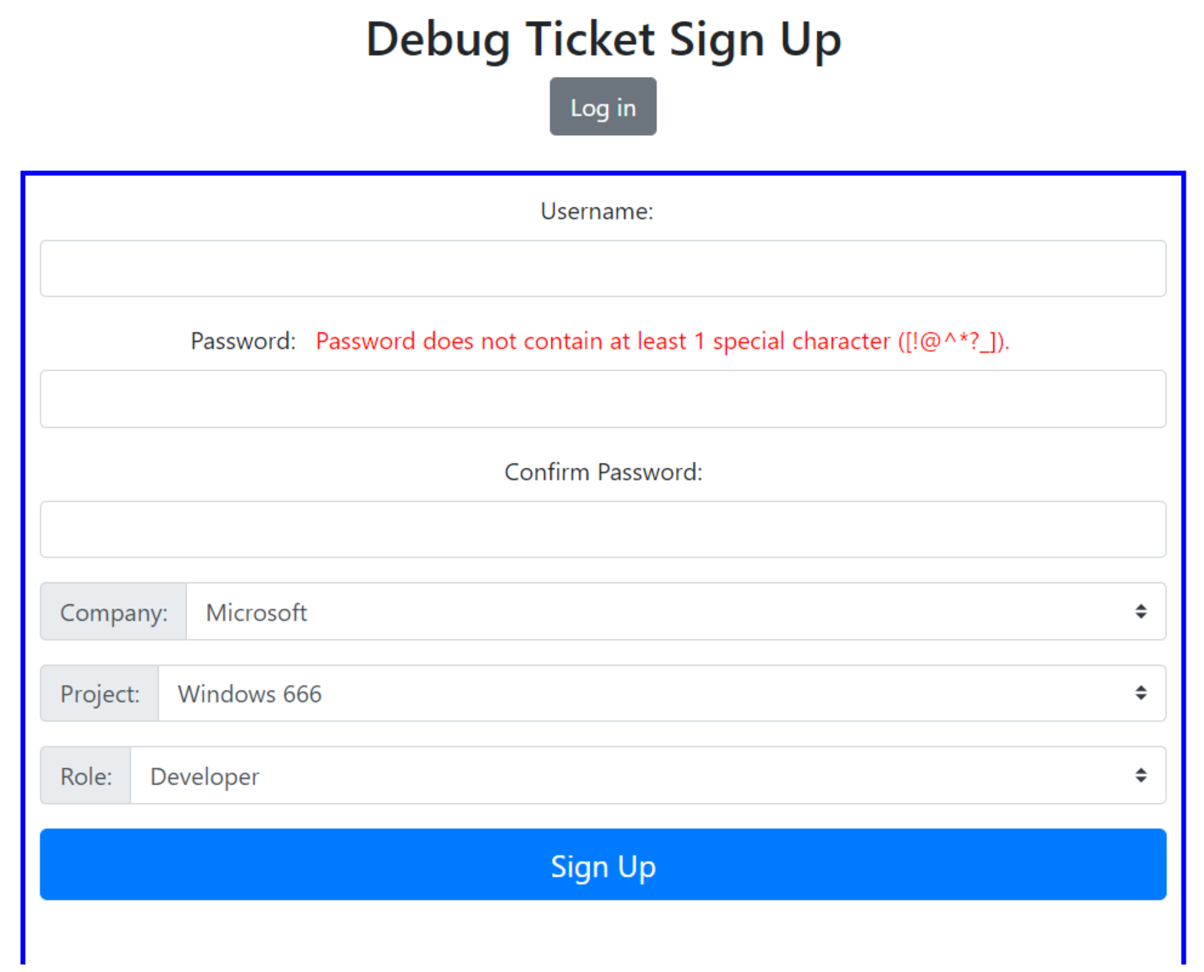
Company: Microsoft

Project: Windows 666

Role: Developer

Sign Up

password not alphanumeric, password has no special character


Debug Ticket Sign Up

Log in

Username:

Password: Password does not contain at least 1 special character ([!@^*?_]).

Confirm Password:

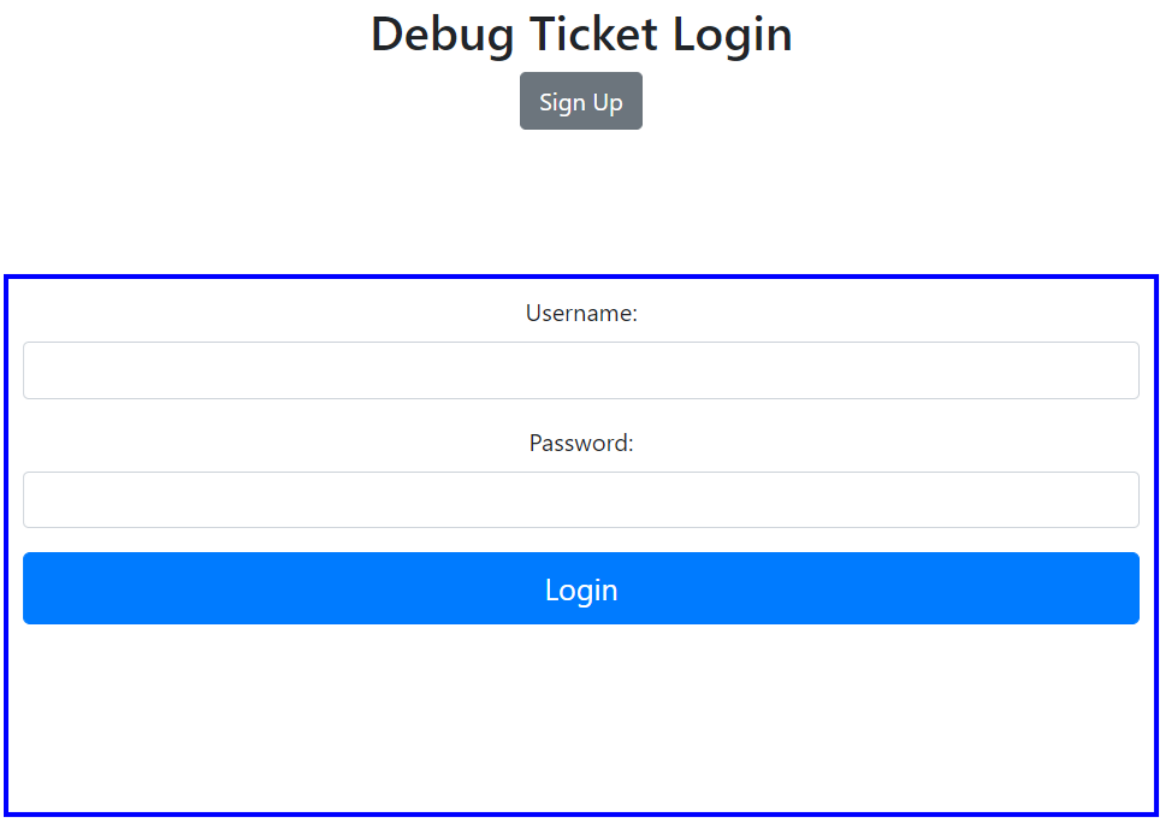
Company: Microsoft

Project: Windows 666

Role: Developer

Sign Up

password has no special character


Debug Ticket Login

Sign Up

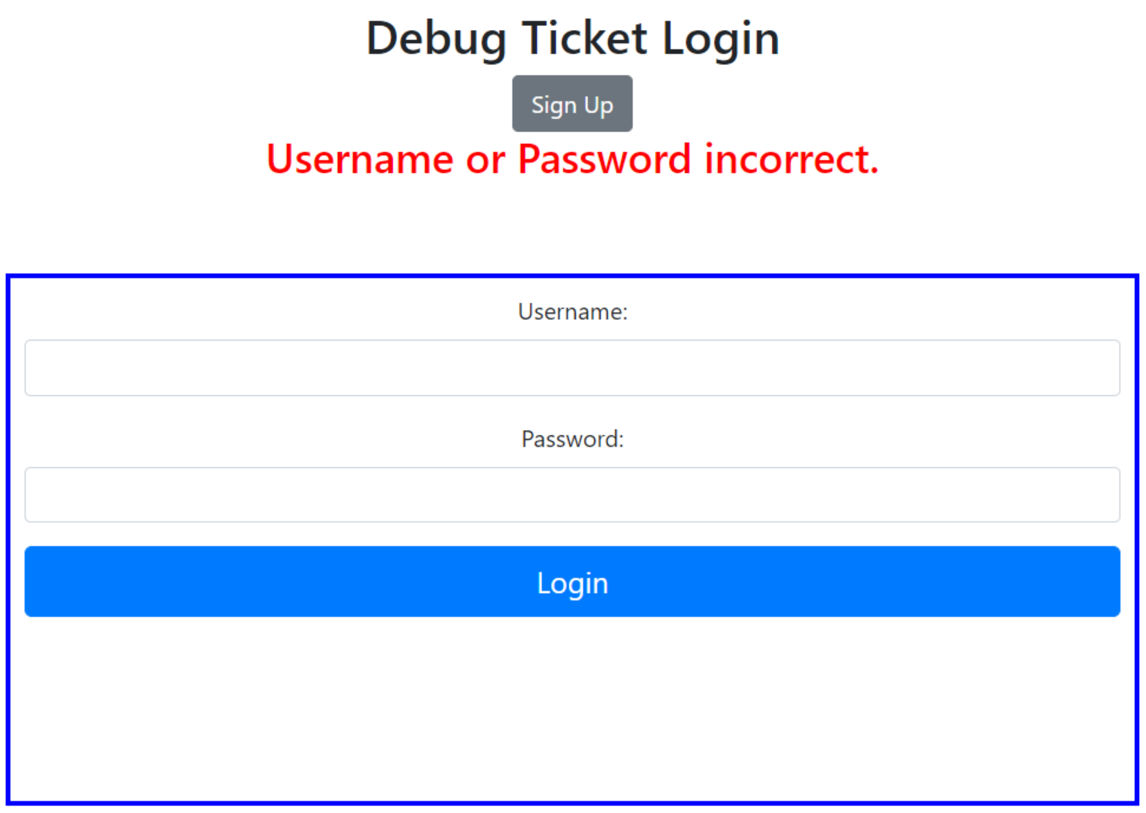
Username:

Password:

Login

Sign up was successful, user directed back to login page

Login



Debug Ticket Login

Sign Up

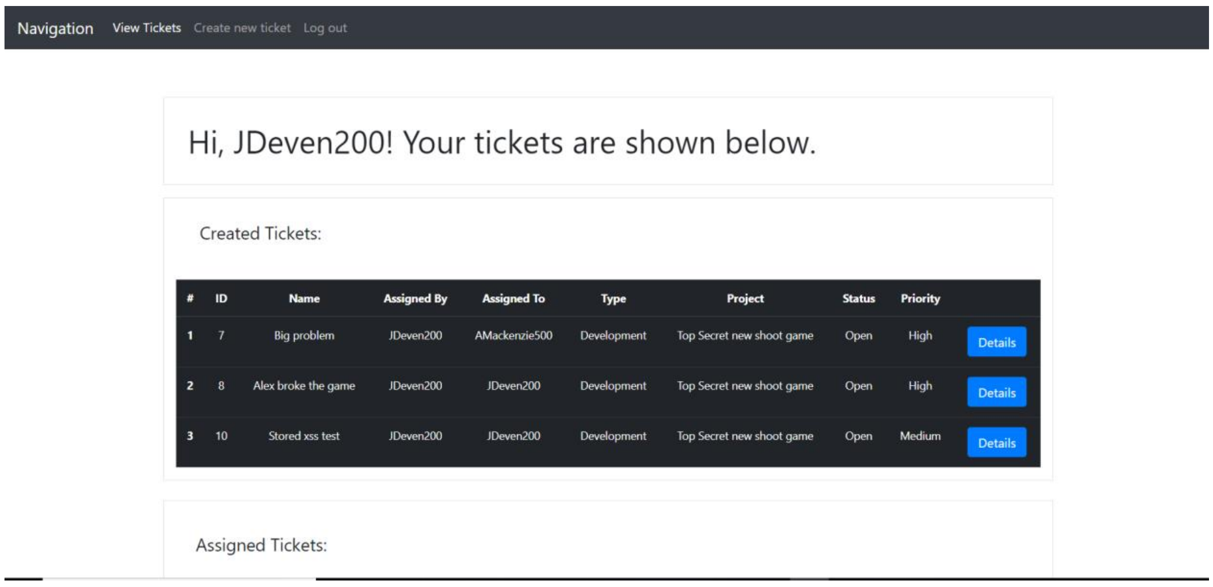
Username or Password incorrect.

Username:

Password:

Login

Username or password incorrect.



Navigation View Tickets Create new ticket Log out

Hi, JDeven200! Your tickets are shown below.

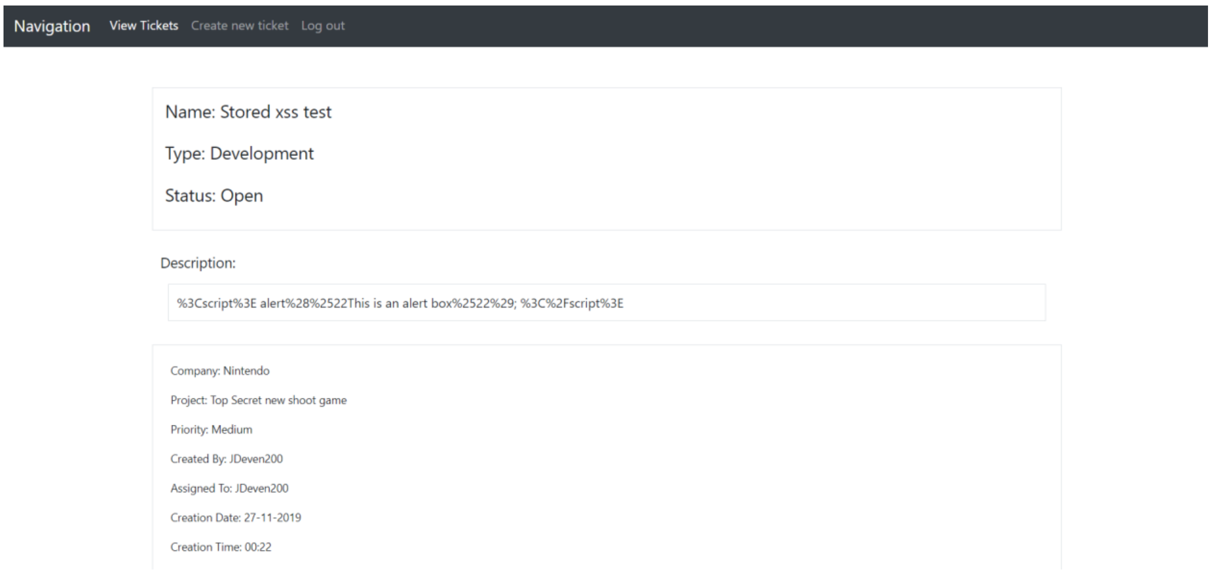
Created Tickets:

#	ID	Name	Assigned By	Assigned To	Type	Project	Status	Priority	
1	7	Big problem	JDeven200	AMackenzie500	Development	Top Secret new shoot game	Open	High	Details
2	8	Alex broke the game	JDeven200	JDeven200	Development	Top Secret new shoot game	Open	High	Details
3	10	Stored xss test	JDeven200	JDeven200	Development	Top Secret new shoot game	Open	Medium	Details

Assigned Tickets:

Username and password correct, directed to the home page. Tickets are loaded based on user ID.

View Ticket



Name: Stored xss test

Type: Development

Status: Open

Description:

```
%3Cscript%3E alert%28%2522This is an alert box%2522%29; %3C%2Fscript%3E
```

Company: Nintendo

Project: Top Secret new shoot game

Priority: Medium

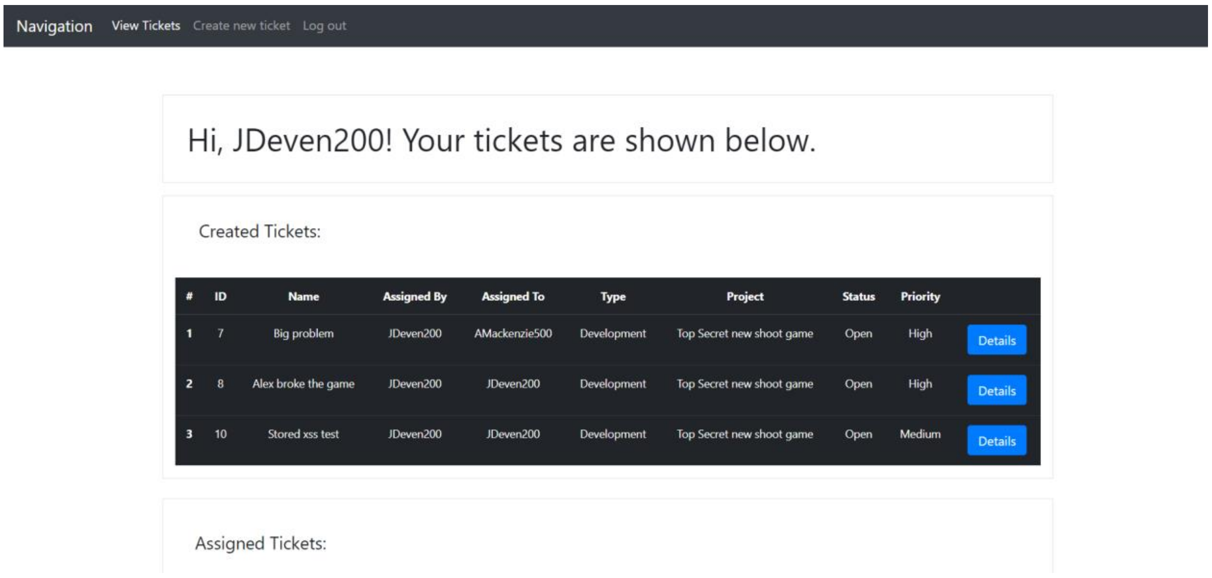
Created By: JDeven200

Assigned To: JDeven200

Creation Date: 27-11-2019

Creation Time: 00:22

Ticket ID (10) matches one of the tickets associated with the currently logged in user. The ticket information is loaded.

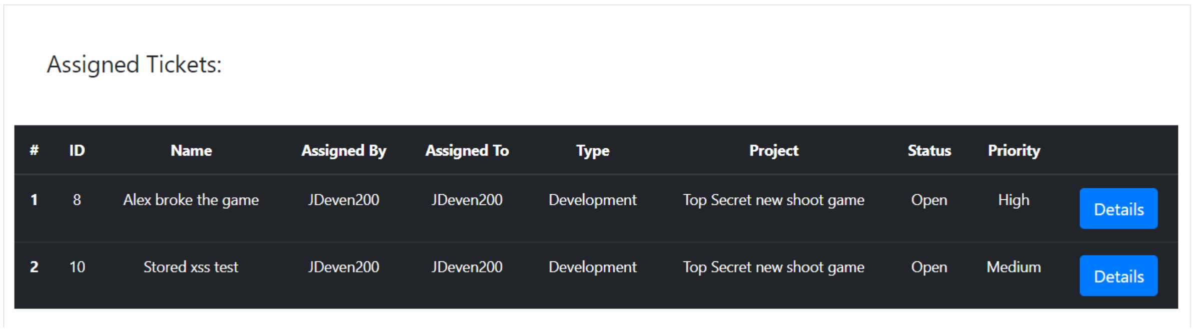


Hi, JDeven200! Your tickets are shown below.

Created Tickets:

#	ID	Name	Assigned By	Assigned To	Type	Project	Status	Priority	
1	7	Big problem	JDeven200	AMackenzie500	Development	Top Secret new shoot game	Open	High	Details
2	8	Alex broke the game	JDeven200	JDeven200	Development	Top Secret new shoot game	Open	High	Details
3	10	Stored xss test	JDeven200	JDeven200	Development	Top Secret new shoot game	Open	Medium	Details

Assigned Tickets:

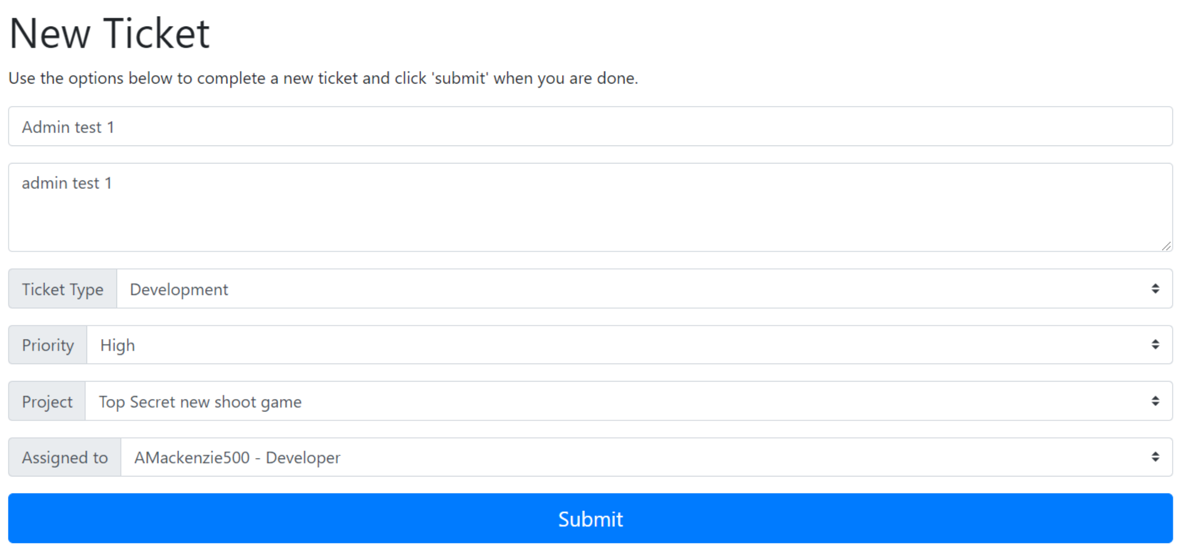


Assigned Tickets:

#	ID	Name	Assigned By	Assigned To	Type	Project	Status	Priority	
1	8	Alex broke the game	JDeven200	JDeven200	Development	Top Secret new shoot game	Open	High	Details
2	10	Stored xss test	JDeven200	JDeven200	Development	Top Secret new shoot game	Open	Medium	Details

Ticket ID (11) does not match any of the tickets created or assigned tickets and the user is booted back to the home screen.

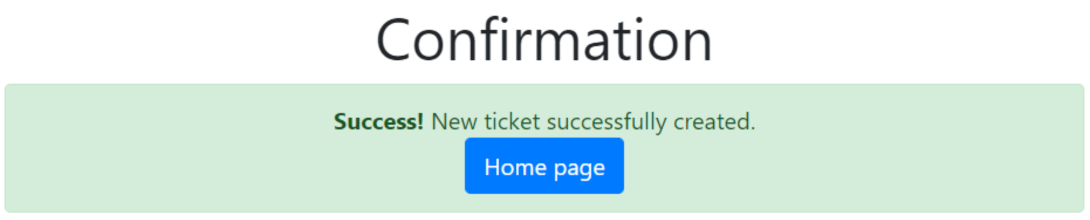
Create Ticket



New Ticket

Use the options below to complete a new ticket and click 'submit' when you are done.

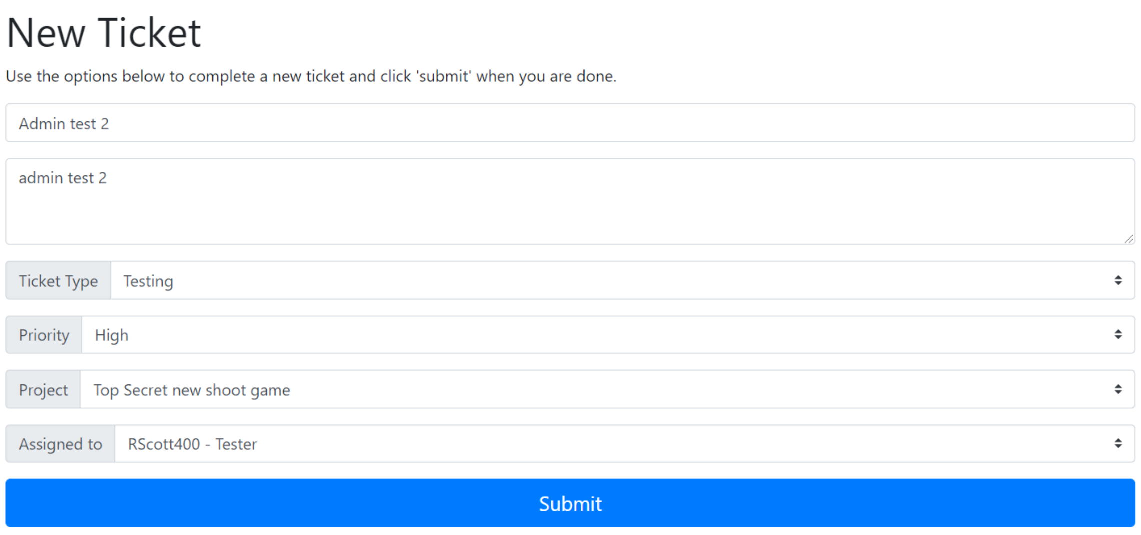
	Admin test 1
	admin test 1
Ticket Type	Development
Priority	High
Project	Top Secret new shoot game
Assigned to	AMackenzie500 - Developer



Confirmation

Success! New ticket successfully created.

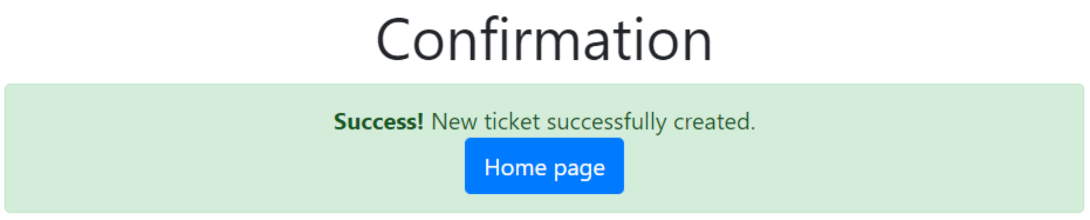
Admin creates a 'Development' ticket



New Ticket

Use the options below to complete a new ticket and click 'submit' when you are done.

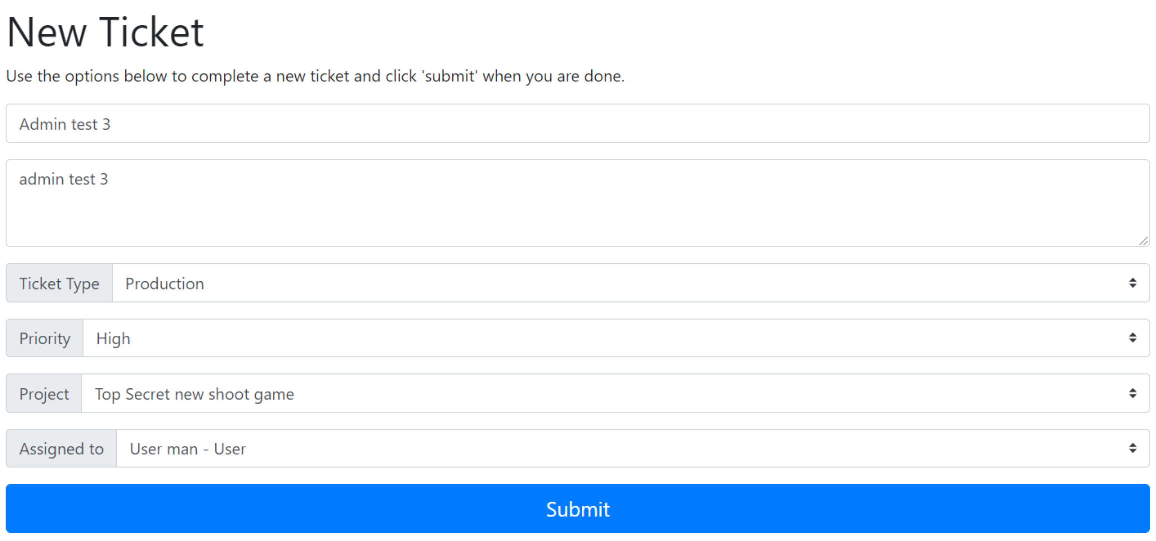
	Admin test 2
	admin test 2
Ticket Type	Testing
Priority	High
Project	Top Secret new shoot game
Assigned to	RScott400 - Tester



Confirmation

Success! New ticket successfully created.

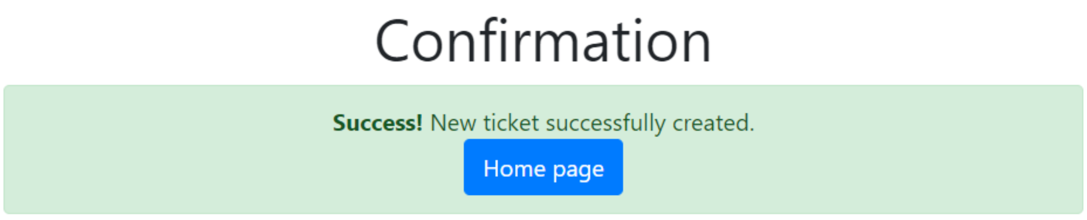
Admin creates a 'Testing' ticket



New Ticket

Use the options below to complete a new ticket and click 'submit' when you are done.

	Admin test 3
	admin test 3
Ticket Type	Production
Priority	High
Project	Top Secret new shoot game
Assigned to	User man - User

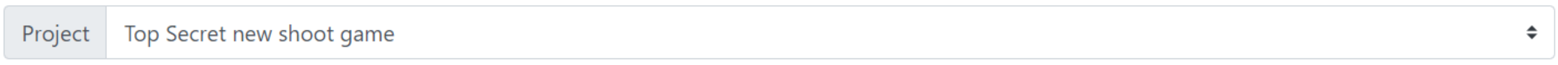


Confirmation

Success! New ticket successfully created.

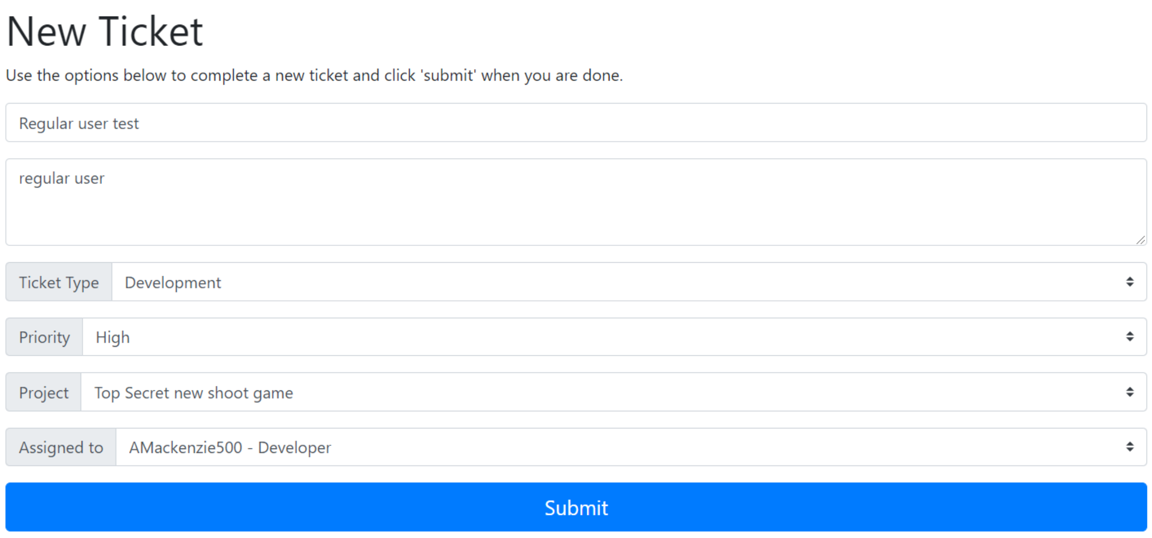
Home page

Admin creates a 'Production' ticket



Project: Top Secret new shoot game

User belongs to the company 'Nintendo', so the user can only view projects which belong to this company



New Ticket

Use the options below to complete a new ticket and click 'submit' when you are done.

Regular user test

regular user

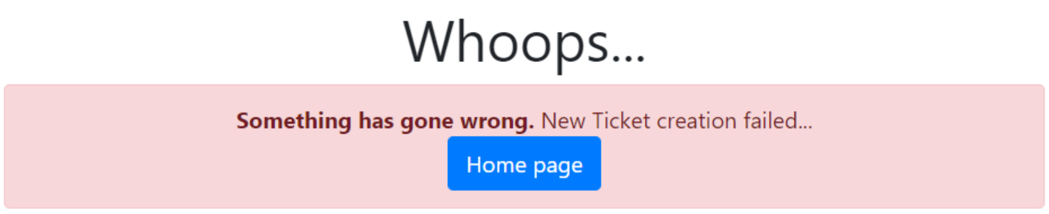
Ticket Type: Development

Priority: High

Project: Top Secret new shoot game

Assigned to: AMackenzie500 - Developer

Submit

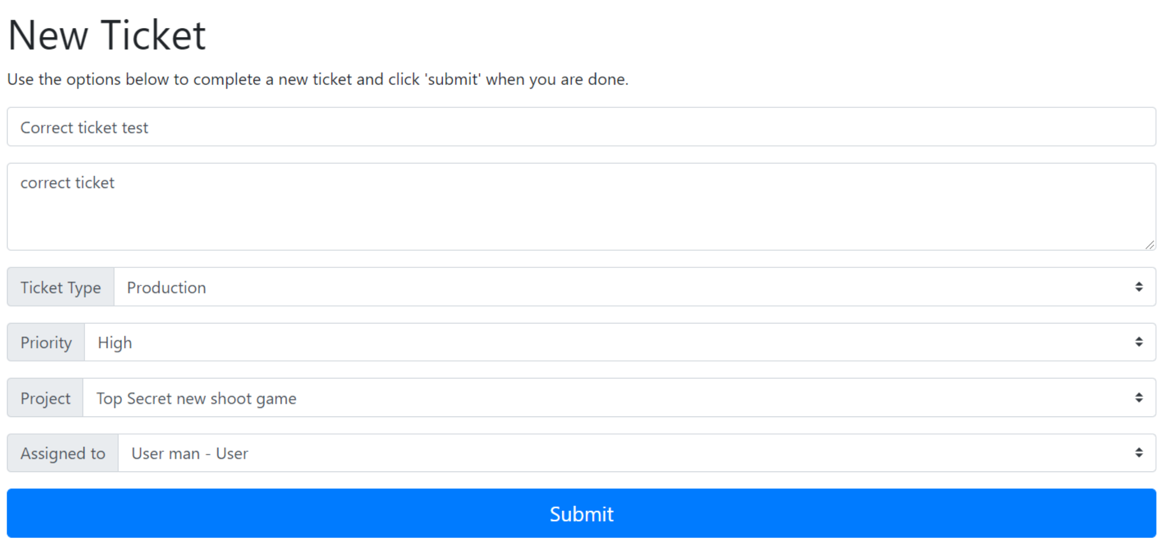


Whoops...

Something has gone wrong. New Ticket creation failed...

Home page

User who is a 'User' in the system (as opposed to 'Developer' or 'Tester') tries to create a 'Development' ticket


New Ticket

Use the options below to complete a new ticket and click 'submit' when you are done.

Correct ticket test

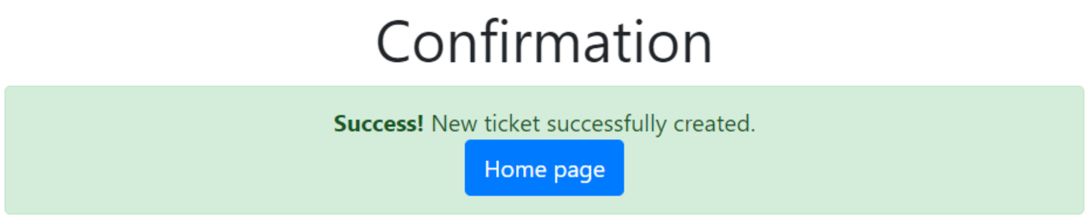
correct ticket

Ticket Type: Production

Priority: High

Project: Top Secret new shoot game

Assigned to: User man - User


Confirmation

Success! New ticket successfully created.

User of type 'User' tries to create a 'Production' ticket