Lab 04-02 Grade 93/100

See comments below where points deducted

horizontal line

# Issue Tracker - Phase 4

This lab is a continuation of the Issue Tracker project that we started previously. In this assignment we will be limiting who is allowed to access the various routes, and take various actions.

This lab assignment will guide you through implementing the following requirements:

1. Users can have roles such as: **Developer, Business Analyst, Quality Analyst, Product Manager, or Technical Manager.**
2. Anonymous users can register for an account, but do not receive a role initially.
3. Technical Managers can edit any user.
4. Technical Managers can assign roles to users.
5. All data is viewable by any role.
6. Any role can create a bug.
   1. Bugs are unclassified initially.
   2. Bugs are assigned to the author initially.
7. Any user can edit a bug that they are the author of.
8. Bugs can also be edited by the user that they are assigned to.
9. Bugs can also be re-assigned by the user that they are assigned to.
10. Business Analysts can edit any bug.
11. Business Analysts and Technical Managers can re-assign any bug.
12. Business Analysts can classify any bug as unclassified/approved/unapproved/duplicate.
13. Business Analysts can close any bug.
14. Any user with a role can add comments to a bug.
15. Quality Analysts can add, edit, execute, and delete test cases for any bug.

# 

# Step 1. Make a table of roles and permissions (10 pts)

Given the information on the previous pages, make a table of the roles and permissions, and denote what permissions each role has with a check mark.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Users** | **DEV** | **QA** | **BA** | **PM** | **TM** |
| canEditAnyUser |  |  |  |  | x |
| canViewData | x | x | x | x | x |
| canAssignRoles |  |  |  |  | x |
| **Bugs** |  |  |  |  |  |
| canCreateBug | x | x | x | x | x |
| canEditAnyBug |  |  | x |  |  |
| canCloseAnyBug |  |  | x |  |  |
| canClassifyAnyBug |  |  | x |  |  |
| canReassignAnyBug |  |  | x |  | x |
| canEditMyBug | x | x | x | x | x |
| canEditIfAssignedTo | x | x | x | x | x |
| canReassignIfAssignedTo | x | x | x | x | x |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| **Comments** |  |  |  |  |  |
| canAddComments | x | x | x | x | x |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| **Test Cases** |  |  |  |  |  |
| canAddTestCase |  | x |  |  |  |
| canEditTestCase |  | x |  |  |  |
| canDeleteTestCase |  | x |  |  |  |

# 

# Step 2. Make a table of permissions and routes (10 pts)

Given the information on the previous pages, make a table of which permission is required to access each route.

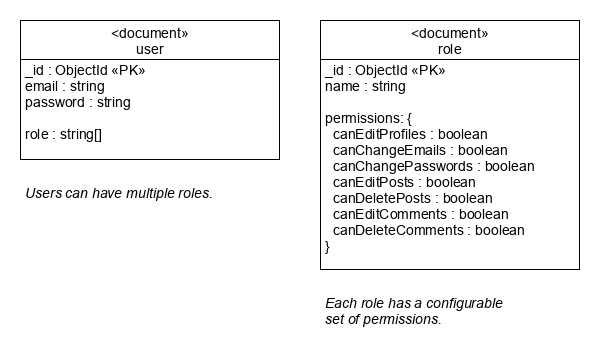
|  |  |
| --- | --- |
| **Route** | **Permission** |
| POST /api/user/register | n/a |
| POST /api/user/login | n/a |
| GET /api/user/list | canViewData |
| GET /api/user/me | n/a |
| GET /api/user/:userId | canViewData |
| PUT /api/user/me | n/a |
| PUT /api/user/:userId | canEditAnyUser |
| DELETE /api/user/:userId | canEditUser |
| GET /api/bug/list | canViewData |
| GET /api/bug/:bugId | canViewData |
| PUT /api/bug/new | canCreateBug |
| PUT /api/bug/:bugId | canEditAnyBug, canEditIfAssignedTo, canEditMyBug |
| PUT /api/bug/:bugId/classify | canClassifyAnyBug |
| PUT /api/bug/:bugId/assign | canReassignAnyBug |
| PUT /api/bug/:bugId/close | canCloseAnyBug |
| GET /api/bug/:bugId/comment/list | canViewData |
| GET /api/bug/:bugId/comment/:commentId | canViewData |
| PUT /api/bug/:bugId/comment/new | canAddComments |
| GET /api/bug/:bugId/test/list | canViewData |
| GET /api/bug/:bugId/test/:testId | canViewData |
| PUT /api/bug/:bugId/test/new | canAddTestCase |
| PUT /api/bug/:bugId/test/:testId | canEditTestCase |
|  |  |
| DELETE /api/bug/:bugId/test/:testId | canDeleteTestCase |

# Step 3. Add roles to database (10 pts)

Take the information in **Step 1,** and add roles to the **roles** collection in your database.

Set the permissions for each role as described in **Step 1.**

Use the example UML diagram on the right to model roles and permissions.

* **Developer**
* **Business Analyst**
* **Quality Analyst**
* **Product Manager**
* **Technical Manager**

# Step 4. Implement authorization middleware functions (4 pts)

Use the example code provided on **pages 15-22** of the lecture notes to implement these authorization middleware functions:

* **isLoggedIn**
* **hasAnyRole**
* **hasRole**
* **hasPermission**

[21 Authorization lecture outline](https://docs.google.com/document/d/1u-L4G9RiBq6qkWbOYvf0ld4OX3wD1nf1n7NyForZac0/edit?usp=sharing)

# 

# Step 5. Update Users API

## POST /api/user/login (5 pts)

1. Update code to add **permissions** to the authPayload. *(See pages 26-27 for how to combine permissions from multiple roles.)*

## POST /api/user/register (5 pts)

1. Update code to add **permissions: {}** to the authPayload. *(User does not start with a role, and therefore has no permissions.)*

## GET /api/user/list (2 pts)

* *Returns info about all users, and supports searching.*
* Do not show passwords.
* Only users with permission ***canViewData*** can access this route.

## GET /api/user/me (2 pts)

* *Returns info about your own account.*
* Do not show passwords.
* Any logged in user can access this route.

## GET /api/user/:userId (2 pts)

* *Returns info about a particular user.*
* Do not show passwords.
* Only users with permission ***canViewData*** can access this route.

## PUT /api/user/me (2 pts)

* Any logged in user can access this route.
* Do not allow the user to change their own role. Note: No points deducted, but probably should check and see if user tried to update a role and send back a message that they cannot change their role

## PUT /api/user/:userId (2 pts)

* Only users with permission ***canAnyEditAnyUser*** can access this route.
* Update the Joi schema to allow the role to be either a string or an array of strings. [(See API Reference)](https://joi.dev/api)

## DELETE /api/user/:userId (2 pts)

* Only users with permission ***canEditUser*** can access this route.

# Step 6. Update Bugs API

## GET /api/bug/list (2 pts)

* Only users with permission ***canViewData*** can access this route.

## GET /api/bug/:bugId (2 pts)

* Only users with permission ***canViewData*** can access this route.

## PUT /api/bug/new (2 pts)

* Only users with permission ***canCreateBug*** can access this route.

## PUT /api/bug/:bugId (5 pts) - 3.5 points

* Users with permission ***canEditAnyBug*** can edit any bug.
* Users with permission ***canEditIfAssignedTo*** can edit bugs that they are assigned to.

Need to do something like this: (permissions.hasOwnProperty("canEditIfAssignedTo") && bug.assignedTo?.\_id === req.auth.\_id)

* Users with permission ***canEditMyBug*** can edit bugs that they are the author of.

Need to do something like this (permissions.hasOwnProperty("canEditMyBug") && bug.author?.\_id === req.auth.\_id)

## PUT /api/bug/:bugId/classify (5 pts)

* Users with permission ***canClassifyAnyBug*** can classify any bug.
* Users with permission ***canEditIfAssignedTo*** can classify bugs that they are assigned to.
* Users with permission ***canEditMyBug*** can classify bugs that they are the author of.

## PUT /api/bug/:bugId/assign (5 pts) - 3.5 points

* Users with permission ***canReassignAnyBug*** can re-assign any bug.
* Users with permission ***canReassignIfAssignedTo*** can re-assign bugs that they are assigned to.

Need to do something like this: (permissions.hasOwnProperty("canReassignIfAssignedTo") && bug.assignedTo?.\_id === req.auth.\_id)

* Users with permission ***canEditMyBug*** can re-assign bugs that they are the author of.

Need to do something like this (permissions.hasOwnProperty("canEditMyBug") && bug.author?.\_id === req.auth.\_id)

## PUT /api/bug/:bugId/close (5 pts)

* Only users with permission ***canCloseAnyBug*** can access this route.

# Step 7. Update Comments API

GET /api/bug/:bugId/comment/list (2 pts)

* Only users with permission ***canViewData*** can access this route.

GET /api/bug/:bugId/comment/:commentId (2 pts)

* Only users with permission ***canViewData*** can access this route.

PUT /api/bug/:bugId/comment/new (2 pts)

* Only users with permission ***canAddComments*** can access this route.

# Step 8. Update Test Cases API

GET /api/bug/:bugId/test/list (2 pts)

* Only users with permission ***canViewData*** can access this route.

GET /api/bug/:bugId/test/:testId (2 pts)

* Only users with permission ***canViewData*** can access this route.

PUT /api/bug/:bugId/test/new (2 pts)

* Only users with permission ***canAddTestCase*** can access this route.

PUT /api/bug/:bugId/test/:testId (2 pts)

* Only users with permission ***canEditTesteCase*** can access this route.

DELETE /api/bug/:bugId/test/:testId (2 pts)

* Only users with permission ***canDeleteTestCase*** can access this route.