SERVER SIDE

Cab 230

Junshao Lin (n9840524)

Introduction

These functions below are implemented by this project

- Use express
- Reponses for information endpoints
- Reponses for authentication endpoints
- Login and register function
- Offence query function
- Filter search function
- Basic search function
- Filter condition (areas, genders, ages, years)
- HTTPS
- Logging (Morgan)
- Swagger

Also, the security aspect such as helmet and JWT is successfully implemented to protect the server and users' information.

Technical description

All requests have been separated into different routers. ‘app.js’ catches a request first and then passing the request into a specific router to deal with the requests.

There are 5 routers to deal with different requests.

- Users router
- Offences router
- Filter condition router
- Search router
- Advance filter router

‘Users’ router deals with account operations such as ‘/login’ and ‘/register’. When a user successfully login or register, the router will response a success message with a new token. This token is used for functions about search.

‘Offences’ router deals with ‘/offences’ query request. Once a user invokes an offences query request, this router will send all offences name to users.

‘Filter condition router’ routers deals with request such as ‘/years’, ‘/ages’, ‘/areas’ and ‘/genders’. Once a user invokes an offences query request, this router will send the data according to user’s request to users.

‘Search’ router deals with basic search request such as ‘/search’. It requires the name of offences as query condition and token to verify the authorization.

‘Advance filter’ router deals with advance search request such as ‘/advanceFilter’. It requires multiple search conditions such as offences, areas, years and gender. It also requires token to verify the authorization.

Security

For the security concerns, this server has implemented these functions that show below

- HTTPS - uses TLS approach to protect the connection and data transition between server and client.
- Password hashing - All password is hashed before they are saved into a database, this operation prevent other people can direct get password from the database.
- Helmet – a security middleware to force clients use https protocol.
- JWT authorization verification to search functions - every time when users invoke a search request, the server will check the authorization first. The search result will send to users if they pass the verification. This operation prevent exposing more data to those users who do not have accounts.
- Knex - Knex is used to prevent SQL injection
- Morgan - to preserve the interactions with the server

Testing and limitations

Test Case	
Test Case ID: L_002	**Test Designed by**: Junshao Lin
Test Priority(Low/Medium/High): High	**Test Designed date**: 25/05/2019
Module Name: Login	**Test Executed by**: Junshao Lin
Test Title: Login functions	**Test Execution date**: 25/05/2019
Description: Verify if server response login successfully	

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Start the server				Pass	
2	Invoke login post request with non-existed account	Email:123@456.com Password:12345678	Code 401 With message: Email or password wrong	Code 401 With message: Email or password wrong	Pass	
3	Invoke login post request with existed email and wrong password	Email:111@111 Password:87654321	Code 401 With message: Email or password wrong	Code 401 With message: Email or password wrong	Pass	
4	Invoke login post request with existed account	Email: 111@111 Password:12345678	Code 200 With token and successful message	Code 200 With token and successful message	Pass	

Pre-Requested: The server side contains an account '111@111' with password '12345678'
Result: All steps are passed. Thus this function is verified as successful

Test Case	
Test Case ID: R_002	**Test Designed by**: Junshao Lin
Test Priority(Low/Medium/High): High	**Test Designed date**: 25/05/2019
Module Name: Register	**Test Executed by**: Junshao Lin
Test Title: Register functions	**Test Execution date**: 25/05/2019
Description: Verify if server response register successfully	

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Start the server				Pass	
2	Invoke register post request with existed account	Email:111@111 Password:12345678	Code 400 With message User already exists	Code 400 With message User already exists	Pass	
3	Invoke register post request with non-existed account	Email:111@333 Password:12345678	Code 201 With message successful and token	Code 201 With message successful and token	Pass	

Requirement: The server side contains an account '111@111' with password '12345678' and the server does not contains an account '111@333'
Result: All steps are passed. Thus this function is verified as successful

Test Case						
Test Case ID: O_001			**Test Designed by**: Junshao Lin			
Test Priority(Low/Medium/High): High			**Test Designed date**: 28/05/2019			
Module Name: Offences			**Test Executed by**: Junshao Lin			
Test Title: Offence functions			**Test Execution date**: 28/05/2019			
Description: Verify if offence function response correctly						
Step	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass/Fail)**	**Notes**
1	Start the server				Pass	
2	Invoke offence get request		Code 200 With message success and a list of offences' name	Code 200 With message success and a list of offences' name	Pass	

Result: All steps are passed. Thus this function is verified as successful

Test Case	
Test Case ID: F_001	**Test Designed by**: Junshao Lin
Test Priority(Low/Medium/High): High	**Test Designed date**: 28/05/2019
Module Name: Filter	**Test Executed by**: Junshao Lin
Test Title: Filter functions	**Test Execution date**: 28/05/2019
Description: Verify if all filter conditions response correctly	

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Start the server				Pass	
2	Invoke areas get request		Code 200 With result that contains all areas' name	Code 200 With result that contains all areas' name	Pass	
3	Invoke genders get request		Code 200 With result that contains all genders' name	Code 200 With result that contains all genders' name		
4	Invoke ages get request		Code 200 With result that contains all ages' name	Code 200 With result that contains all ages' name		
5	Invoke years get request		Code 200 With result that contains all years' name	Code 200 With result that contains all years' name		

Result: All steps are passed. Thus this function is verified as successful

Test Case	
Test Case ID: S_002	**Test Designed by**: Junshao Lin
Test Priority(Low/Medium/High): High	**Test Designed date**: 01/06/2019
Module Name: Search	**Test Executed by**: Junshao Lin
Test Title: Filter functions	**Test Execution date**: 01/06/2019
Description: Verify if all filter conditions response correctly	

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Start the server				Pass	
2	Invoke search get request with token and an offence name	Offence: Arson	Code 200 With result list that with element like {LGA: a, total: 1, lat: 1, lng: 2}	Code 200 With result list that with element like {LGA: a, total: 1, lat: 1, lng: 2}	Pass	
3	Invoke search get request with token and an offence name	Offence: Arm	Code 400 With message: parameters wrong	Code 400 With message: parameters wrong		
4	Invoke advance search get request with token and conditions	Age: adult Areas: Aurukun Shire Council Gender: Female Year1: 2001 Year2: 2009	Code 200 With result contains nothing	Code 200 With result contains nothing		
5	Invoke advance search get request with token and conditions	Age: adult Areas: Banana Shire Council Gender: Female Year1: 2001 Year2: 2009	Code 200 With result contains two results	Code 200 With result contains two results		

Result: All steps are passed. Thus this function is verified as successful

References

Bcryptjs. (2017). Retrieved from https://www.npmjs.com/package/bcryptjs

Helmet. (2019). Retrieved from https://www.npmjs.com/package/helmet

Knex. (2019). Retrieved from https://www.npmjs.com/package/knex

swagger-ui-express. (2019). Retrieved from https://www.npmjs.com/package/swagger-ui-express

Appendix

Installation Guide

1. Change the database setting in '/database/knexconfig.js'
2. Change the openssl key setting at app.js
3. Execute 'npm install' with Terminal or CMD in the root directory of this server
4. After finishing installing, execute 'npm start' with Terminal or CMD in the root directory of this server to start the server.