#### Introduction

This is the server slide report for CAB230 assessment. This server slide application has implemented all requirement. The table below shows the completion of the application

End point	states
/register	DONE
/login	DONE
/search	DONE
/offences	DONE
/areas	DONE
/ages	DONE
/genders	DONE
/years	DONE
Non-functional Requirement	States
https	DONE
Password hashed	DONE

### Technical description

Technical	Description
NodeJs	JavaScript environment http server
Express	Nodejs web framework for building API
Sequelize	Nodejs promise-based ORM
Bcrypt	Password hashing function for protecting
	passowords
Helmet	Express HTTPS security header. Increase the
	security of the express
Json web token (JWT)	RFC 7519 standard authorization token
	signature signed by private key
Mysql	Open-source relational database management
	system

#### Security

```
Technical
                                             Description
JWT Token
                                              HEADER: ALGORITHM & TOKEN TYPE
This system used JWT token for
                                               "alg": "HS256",
"typ": "JWT"
securely transmit information
between front-end and backend. It is
                                              PAYLOAD: DATA
hashed with HS256 algorithm signed
with the secret code.
                                               "sub": "1234567890",
With the JWT token, it can
                                               "name": "John Doe",
                                               "iat": 1516239022
successfully prevent information
integrity, even the data with in the
                                              VERIFY SIGNATURE
JWT token is public to anyone who
                                             HMACSHA256(
                                              base64UrlEncode(header) + "." +
gets the token.
                                              base64UrlEncode(payload),
                                              your-256-bit-secret
                                             secret base64 encoded
```

Hashed password	email	password	T
The photo on the right hand side is the user database table, it shows the email and password. The password in the image is in an hashed format, this format is called bcrypt password hashing with salt password, which has a strong protection against rainbow table attacks.	w@gmail.com jeffreylau1129@gmail.com w2@gmail.com jeffreylau11239@gmail.com jeffreylau112349@gmail.com jeffreylau112349@gmail.com awef@wef.com aefawefa@wefcom.i wefwef#werfer/#@c	password  \$2b\$04\$zNZRxOwH7Gn7cTVI44aeNeoFgWKVV  \$2b\$04\$OoWa4Y4J.No5DW3/zI81juMvsJgHF/e  \$2b\$04\$FKqueArWkexlNtOTI7G1FOWonq.7Q9  \$2b\$04\$fTq.nR7bHLTH8nD18SDEheV1B/7Rsfik  \$2b\$04\$fGtaBVTGBxSssW8SHh4/.Az3TFcwjHC  \$2b\$04\$ZCJazxk2qwcITq4zV3ft2.GH8ICLhhx0i  \$2b\$04\$dx0BaTQSUo0.3/xwdm0neeoDMcmBz1  \$2b\$04\$HKFa7y8G75OqQFoZeZSbeVgI7ZaT7o	
Helmet header Helmets provides some various of https header protection for the express sever which prevent DNS prefetching, clickjacking, remove X- Power,Xss protection and others basic security protection		equire('swagger-ui-expre: nt = require('./docs/swa <sub>l</sub> ;	
https Https namely HyperText Transfer Protocol Secure that provide extended security protection for http. It is commonly used nowadays to keep information communication securely.	* General  Request URL: https://	172.22.30.8:8443/offences	

## Testing and limitation

For testing the API, the server side has implemented swaggerJS in order to try out the api result manually. The testing swagger testing link is: <a href="https://172.22.30.8:8443/">https://172.22.30.8:8443/</a>

Testing Scenario: Register

Test Case	Request	Response	Result
Register	Request URL:	Code status:	CORRECT
with a new user email	https://172.22.30.8:8443/register	201	
	Request body:	Response body:	
	<pre>{     "email":"newUsesr@gmail.com",     "password":"demouserpassword" }</pre>	<pre>{    "message": "yay! you've successfully registered your user account :)" }</pre>	
Register a	Request URL:	Code status:	CORRECT
existing user email	https://172.22.30.8:8443/register	400	
	Request body:	Response body:	
	{	{	
	"email":"newUsesr@gmail.com",	"message": "oops! It looks	
	"password":"demouserpassword"	like that user already	
	}	exists :("	

		_	
		l	
		J	
		_	
1	1		1

### esting Scenario: Login

Test Case	Request	Response	Result
Login with an existing user email	Request URL: https://172.22.30.8:8443/login  Request body: {     "email":"newUsesr@gmail.com",     "password":"demouserpassword" }	Code status: 200  Response body: {     "token":     "eyJhbGciOiJIUzI1NiIsInR5cCI6Ik     pXVCJ9.eyJ1c2VyIjp7ImlkIjo2LCJI     bWFpbCI6Im5ld1VzZXNyQGdtYWlsLmN     vbSJ9LCJpYXQiOjE1NTkyMTI2MTksIm     V4cCI6MTU1OTI5OTAXOXO.itV5MMWB9     H0Fxtn7NQKFobBbxue7PPsrqCnZ5uF4     1Gs",          "access_token":     "eyJhbGciOiJIUzI1NiIsInR5cCI6Ik     pXVCJ9.eyJ1c2VyIjp7ImlkIjo2LCJl     bWFpbCI6Im5ld1VzZXNyQGdtYWlsLmN     vbSJ9LCJpYXQiOjE1NTkyMTI2MTksIm     V4cCI6MTU1OTI5OTAXOXO.itV5MMWB9     H0Fxtn7NQKFobBbxue7PPsrqCnZ5uF4     1Gs",          "token_type": "Bearer",          "expires_in": 86400     }	CORRECT
Login with Wrong user detail	Request URL: https://172.22.30.8:8443/login	Code status: 401	CORRECT
	<pre>Request body: {     "email":"newUsesr@gmail.com",     "password":"wrongpassword" }</pre>	<pre>Response body: {     "message": "invalid login - bad password" }</pre>	

#### Testing Scenario: Help

Test Case	Request	Response	Result
Get all	Request URL:	Code status:	CORRECT
offences	https://172.	200	
	22.30.8:8443	Response body:	
	/areas	<pre>{    "offences": [      "Advertising Prostitution",      "Armed Robbery",      "Arson",      "Assault",      "Attempted Murder",     ]</pre>	
Get all	Request URL:	Code status:	CORRECT
Areas	https://172.	200	COMMECT
	22.30.8:8443		
	/areas	Response body:	
	,	{	
		"areas": [	
		"Aurukun Shire Council",	
		"Balonne Shire Council",	
		"Banana Shire Council",	
		"Barcaldine Regional Council",	

```
"Barcoo Shire Council",
                                    ]
                              }
Get all
             Request URL:
                                                                                         CORRECT
                              Code status:
             https://172.
ages
                              200
             22.30.8:8443
                              Response body:
             /ages
                              {
                                   "ages": [
   "Adult",
                                     "Juvenile"
                               }
Get all
             Request URL:
                              Code status:
                                                                                         CORRECT
             https://172.
22.30.8:8443
genders
                              Response body:
             /genders
                                   "genders": [
                                     "Female",
                                     "Male",
                                     "Not stated",
                                   ]
                                 }
Get all
             Request URL:
                              Code status:
                                                                                         CORRECT
years
             https://172.
             22.30.8:8443
                              Response body:
             /years
                                   "years": [
                                     2001,
                                     2002,
                                     . . .
                                 }
```

Testing Scenario: Search

Test Case	Request	Response	Result
Test Case Search with validate JWT token	Request URL: https://172.22.30.8:8443/ search?offence=Homicide(Murder)& area=Banana%20Shire%20Council& age=Adult&gender=male& year=2001&month=2	Response  Code status: 200  Response body: {     "query": {         "offence": "Homicide (Murder)",     "gender": "male",     "year": "2001",     "age": "Adult",     "area":	Result
		<pre>"area": "Banana%20Shire%20Council",      },      "result": [           {                 "LGA": "Aurukun Shire Council",                 "total": 4,                 "lat": -27.470812,                  "lng": 153.022455           }          </pre>	

		}	
search without offence query parm	Request URL: https://172.22.30.8:8443/ search	<pre>Code status: 400  Response body: {     "message": "oops! it looks like you're missing the offence query parm"     } }</pre>	CORRECT
search with invalidate JWT token	Request URL: https://172.22.30.8:8443/ search?offence=Homicide(Murder)& area=Banana%20Shire%20Council& age=Adult&gender=male& year=2001&month=2	<pre>Code status: 401  Response body: {     "message": "oh no! it looks like your authorization token is invalid"     }</pre>	CORRECT

### References

Technical	url
NodeJs	https://nodejs.org/en/
Express	https://expressjs.com/
Sequelize	https://www.npmjs.com/package/sequelize
Bcrypt	https://www.npmjs.com/package/bcrypt
Helmet	https://helmetjs.github.io/
Json web token (JWT)	https://jwt.io/
Mysql	https://www.mysql.com/

# Appendix

Instruction for running the application

- 1. npm install
- 2. npm run start