Documentation of CSC 301

Team 19

April 10, 2020

Contents

1 Technical Structure

1.1 Backend

- Python using a library called Flask
- Database is MongoDB
  - Emails stored in the database are normalized to ensure a consistent comparable format. What this means is hello@GMAIL.com gets converted to hello@gmail.com and so on. There is an example on how this works at the end of this document under Code Samples in case you wish to expand/change the backend.

1.2 Frontend

- React Application (a library available for Node.js)

1.3 Security

- Passwords are hashed and salted using a tool called bcrypt
  - The password stored in the database contains both the hashed and salted password as well as the salt. Bcrypt knows how to isolate the salt.
  - This is theoretically very hard to reverse so if your database is leaked, it is harder to attack it to actually find out the passwords.
- Input validation on the data from the frontend should be covering all routes
- Npm will try to resolve some vulnerabilities. It is unable to fix all of them and you may wish to look into these to ensure that the website is secure in the long term.
- Some of the Python packages have hardcoded versions. You may wish to update these in the future to prevent vulnerabilities from forming. You can change this in main-project/requirements.txt
  - This is also where you can add new Python packages to be installed in case you wish to change/expand the backend.

2 Configuration

- The configuration for the backend is located at main-project/app/config.py
- Configuration should be verified before deployment

2.1 Variables and what they do

2.1.1 Configuration Variables

These variables are located in main-project/app/config.py

SECRET_KEY This is the secret key for the session cookie, an encrypted cookie stored on the browser of the user. If someone knows this key, they can modify their own cookie and authenticate as another user. **Do not show this publicly and edit this after accessing the source code to prevent us from being able to do malicious things.**

STATIC_FOLDER Do not change this, this points to the React app so that the Flask server knows where to go to for 'static' HTML files.

ENABLE_DEBUG_ROUTES When True, this opens up debug routes to the public. **This should be set to False for all production deployments**. When this is set to False, those routes will return a 404 instead (therefore not appearing to be a real route to the general public)

DBUSER This is the username for the administrator account on the MongoDB instance. **Do not show this publicly and edit this after accessing the source code to prevent us from being able to do malicious things.**

DBPASSWORD This is the password for the administrator account on the MongoDB instance. **Do not show this publicly and edit this after accessing the source code to prevent us from being able to do malicious things.**

MONGOURI This tells the server where the MongoDB instance is. **This should be changed once you determine where your production MongoDB server will be.**

DATABASE This points to the **database** within a MongoDB instance. The idea is instance -> **database** -> collection (eg. reports, users) ->

record. Our provider only allowed us one **database** with an unlimited amount of collections within that database. Once you determine what your database will be named, change this variable to be the same.

2.1.2 Other Variables

session This is a special variable associated with the Flask app. It allows you to modify the session cookie and functions as a dictionary unique to each computer connecting to the server.

3 Hosting Recommendations

- We recommend hosting the site on a subdomain of mcode.club, eg. marks.mcode.club
- As to hosting it, the site is run through a Docker image and is relatively portable. Many online services such as Heroku, AWS, etc. allow you to host Docker images.
- We have no specific recommendations for MongoDB providers, Heroku integrates with a provider called mLab which is the service we used for our demo.
- We recommend ensuring that your host supports TLS/SSL (ie. that someone can connect through https so that way passwords can be sent safely from the user to the server)

4 Deployment

4.1 Requirements

- You must have several packages/pieces of software installed
  - Node.js and NPM (used to build react) Link
  - Docker (used to run a medium weight Linux distribution containing the web server, deployable on Heroku, AWS, etc.) Link
  - Make
  - A shell of your choice (eg. bash)
  - Heroku CLI (optional, only if you are deploying to Heroku, installable through NPM)

    * You can run **make getheroku** in the main-project directory to install it if you have NPM and make installed. Please have an account setup before running the command.

4.2 First Deployment Pre-Steps

1. Install required software
2. If deploying to Heroku, log in with the CLI
3. Configure main-project/app/config.py appropriately. (see 2.1.1)
4. Add an admin user to the database. To do this, you want user-Type to equal 1, and you can use the following text for a password "$2y$12$XC7w31h31eigbig.KDcSWejGq6mFPawyiwvRCqf8cCVpuSd5FGiAC" (without the quotations). This is a hashed and salted version of "password"
    - Alternatively, you can temporarily turn ENABLE_DEBUG_ROUTES to True and go to the route /addsampleuser/admin after deployment. This will create a user with the email admin@mcode.club with the password "I love rock and roll". Afterwards, change the user's password and disable debug routes.

4.3 Process

- Assuming that you are running bash, are at the directory that this file is located in, and you are deploying to Heroku, you want to do the following things
- Edit main-project/Makefile's entry for shipprod. Right now it says the following thing. You will want to change the **bolded areas** to be the same as your **Heroku App Name**

shipprod:
heroku container:login
heroku container:push web –app **mcc-prod-301**
heroku container:release web –app **mcc-prod-301**

- Afterwards, again assuming that you are in the directory where this file is located, you want to run the following command.

```
cd main-project; make prod
```

- The first deploy will require a large amount of downloading/uploading. This is because it will download a medium weight Linux image (required for packages like bcrypt which needs gcc) as well as install the Python requirements before uploading.

4.4 Troubleshooting (Common Errors we had while developing)

4.4.1 When I browse to the page, it says 404 not found even though it worked before

This is likely due to React not building correctly. Check the build for errors.

4.4.2 Docker container already in use?

Run the following make command to remove the loaded Docker image. Afterwards, you may attempt to deploy as normal.

```
make clean
```

4.4.3 Error with either shipprod or shipdev

Heroku timed out. You essentially need to try again and then clean the container. This means running...

```
# Use shipdev if you are using the dev location
make shipprod; make clean
```

5 Code Samples

- Here you can find some code samples explaining some of the potentially harder to intuitively understand concepts in case you wish to change/expand the backend.

5.1 Email Normalization

- This ensures a consistent email format for comparisons in authentication/other uses on the backend. For example, hello@GMAIL.com changes to hello@gmail.com

```
email = mailsane.normalize(request.json['email'])
if email.error: # if there was an error in normalization
    abort(400)  # Return a 400 (Bad Request)

if dbworker.validateCredentials(str(email), request.json['password']):
    userType = dbworker.getUserType(str(email))
    # str(email) is either the normalized email or the error message
    # resulting from normalization. This includes if a domain name
    # is not valid.
```

6 Authors and Licensing

- Written by Nathan Fischer, Andriyan Bilyk, Steffy Lo, Philip Smith, Dragan Soso, Ari Truanovsky, and Edwin Chan
- Licensed under the MIT License, located in LICENSE in this directory.