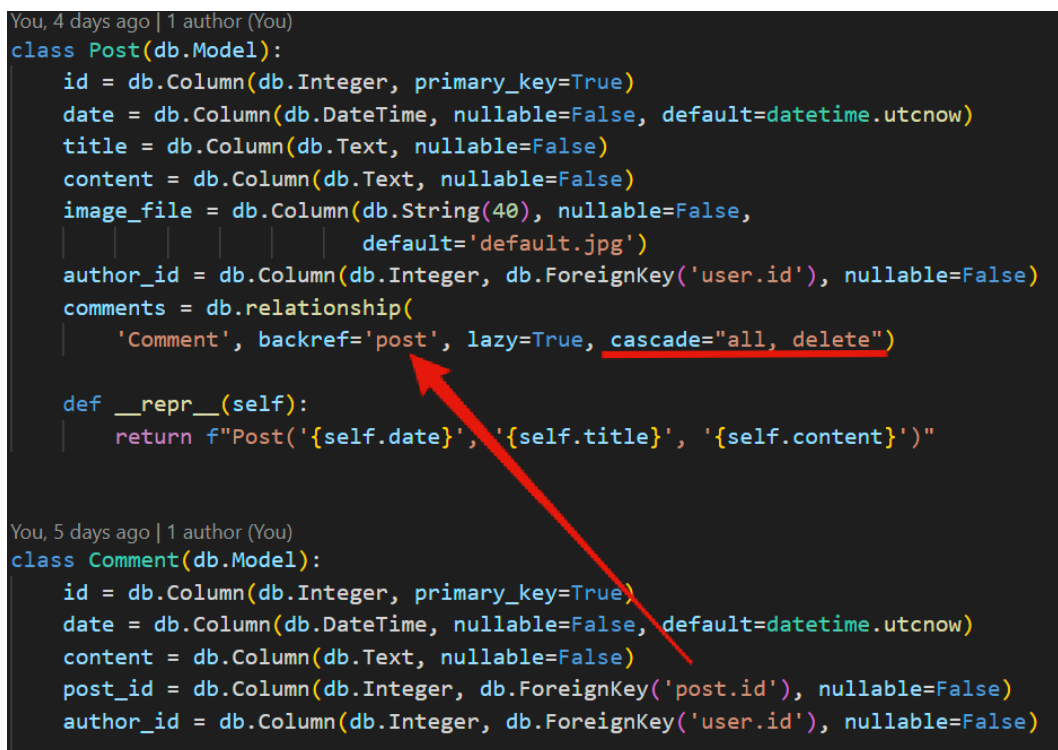


The OpenShift Deployment URL: <http://c22076452-cmt120-cw2-prod-c22076452-cmt120-cw2.apps.openshift.cs.cf.ac.uk/>

Student Number: C22076452

### Quality – Storing Information

The blog uses a MySQL database on deployment and an SQLite file on local testing. The data that the website handles include post and comment texts, JSON objects and image files. Not all data is treated equal as some of them require additional steps for proper processing. Passwords, for example, need to be hashed before being stored on the database as a VARCHAR field, rather than be stored in plaintext form. But an even more important example are images. Due to their significantly increased size (orders of magnitudes bigger than VARCHARs, actually), it would not be efficient to store an image in its full form onto the database. (mrswats, 2021) (davidism, 2015) The standard practice is to use a file system of some sort that would store the images of the user under a randomly generated UUID as a filename (or any sort of randomized string padded onto the original filename) to each uploaded image file in the process to avoid filename collision, then store the filename itself, or its directory, into the database as the image reference for future queries.



```
You, 4 days ago | 1 author (You)
class Post(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    date = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)
    title = db.Column(db.Text, nullable=False)
    content = db.Column(db.Text, nullable=False)
    image_file = db.Column(db.String(40), nullable=False,
                           default='default.jpg')
    author_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    comments = db.relationship(
        'Comment', backref='post', lazy=True, cascade="all, delete")

    def __repr__(self):
        return f"Post('{self.date}', '{self.title}', '{self.content}')"

You, 5 days ago | 1 author (You)
class Comment(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    date = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)
    content = db.Column(db.Text, nullable=False)
    post_id = db.Column(db.Integer, db.ForeignKey('post.id'), nullable=False)
    author_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
```

The image shows a code editor with two Python classes, `Post` and `Comment`, defined using the `db.Model` class. The `Post` class has attributes `id`, `date`, `title`, `content`, `image_file`, `author_id`, and `comments`. The `comments` attribute is a `db.relationship` to the `Comment` class, with `backref='post'`, `lazy=True`, and `cascade="all, delete"`. The `Comment` class has attributes `id`, `date`, `content`, `post_id`, and `author_id`. A red arrow points from the `cascade="all, delete"` line in the `Post` class to the `post_id` attribute in the `Comment` class, indicating the one-to-many relationship and cascading deletion.

Figure 1: A one-to-many relationship between a post and its comments with cascading deletion.

Another aspect of information handling is the deletion of data upon the request of the user. The user may choose to delete comments from posts, delete a post of their own, or even delete their entire account. Since posts contain comments, it forms a one-to-many relationship between them, meaning that if the post was deleted, the comments should also be deleted. Comments and posts form two separate tables on the database, removing a post without removing their corresponding comments would leave those comments in an orphaned state i.e. from a one-to-many to a none-to-many relationship.

The goal here is to define a relationship that forces deletions of all children entries when their singular parent is deleted. In this case, the parent's relationship is specified to delete all children records in a cascading manner as shown in Figure 1, using the "cascade" parameter. The explicit definition of relationships here demonstrates the R in RDBMS, a Relational Database Management System. This very same behaviour applies to user account deletion, whereby posts and comments associated with the user would be deleted as well.

The current implementation of the file upload system uses the static folder to store and manage files uploaded by the user. A major concern stems from the fact that user data is deleted on every new build. A better implementation would be to use an external file system, such as a Content Delivery Network (CDN) to deliver image files on the fly. An inquiry was made about this on the university's StackOverflow, but no response was given suggesting a method that leverages on-site facilities.

### Security – The use of environmental variables to self-configure the app.

The assignment's tutorial handouts provided by the university has suggested that students replace the "SQLALCHEMY\_DATABASE\_URI" field of the application's configuration when migrating from the local SQLite file to the university's MySQL server. Although that method would be the most straightforward and easy-to-implement method, it would expose the credentials for the MySQL server within the source code. And as such, should the repository ever be compromised, the credentials would be compromised along with it. As a preventative measure for this collateral damage, database parameters, including address and credentials, are loaded as a set of environment variables to be read by a custom configuration script.

Project: c22076452-cmt120-cw2 ▼

---

### Data

ENV\_TYPE

.....

MYSQL\_ADDRESS

.....

MYSQL\_DB\_NAME

.....

MYSQL\_PASSWORD

.....

MYSQL\_USER

.....

Figure 2: Environmental variables fields inside a Secrets form.

On OpenShift, the university's containerization suite of choice, credentials may be stored in the Secrets tab in the form of a set of keys and values, as shown in Figure 2. Doing so would enable the application of these secrets directly into a deployment's Environment tab, i.e., all containers under the same deployment would have said secrets initialized automatically as environment variables inside each of them. This is shown in Figure 3.

Project: c22076452-cmt120-cw2 ▼

---

[Deployments](#) > [Deployment details](#)

**D c22076452-cmt120-cw2-prod**

---

Details Metrics YAML ReplicaSets Pods Environment Events

---

Container: **C c22076452-cmt120-cw2-prod** ▼

Single values (env) ⓘ

Name

⋮

[+ Add more](#) [+ Add from ConfigMap or Secret](#)

---

All values from existing ConfigMaps or Secrets (envFrom) ⓘ

CONFIGMAP/SECRET

⋮ **S prod-secrets** ▼

[+ Add all from ConfigMap or Secret](#)

*Figure 3: Linking the secret to a deployment to complete the server-side setup.*

The configuration script would use the key ENV\_TYPE upon start up to check if the current pod is of the Production type, a Staging type, or a Development type. This is because each environment has its own separate set of parameters.

This implementation also allows for the creation and/or usage of the remote databases using a locally run Flask application by using the host machine's own environmental variables, where ENV\_TYPE value would be used to target the appropriate remote database, e.g., PROD for Production type environment, which would create the Production database on the university's MySQL server. This would make it much more efficient when redeploying the application with updated database models throughout development, where the database would need to be reset or migrated quickly.

Since credentials must be available to the system in the form of environmental variables, unauthorized users running on privileged access (su/admin) would be able to read the credentials in plaintext. Therefore, using encrypted passwords instead would make it harder for bad actors to gain access to the database. (Carter, 2012)

## References

Carter, J., 2012. *Is it secure to store passwords as environment variables (rather than as plain text) in config files?*. [Online]

Available at: <https://stackoverflow.com/a/12461944>

[Accessed 17 January 2023].

davidism, 2015. *Serve image stored in SQLAlchemy LargeBinary column*. [Online]

Available at: <https://stackoverflow.com/a/31858076/11690953>

[Accessed 15 January 2023].

mrswats, 2021. *how to store images in sqlalchemy database and display it on React frontend?*. [Online]

Available at: <https://www.reddit.com/r/flask/comments/rdv0m0/comment/ho3kryo/>

[Accessed 15 January 2023].

## Appendix A – Advanced Functionalities

- Cascading deletion relationships (e.g. deleting a post will wipe out all associated comments).
- File upload – for avatars and post images.
- File size check (8MB limit for post images, 2MB for user avatars)
- Uploaded file deletion upon post deletion.
- User-specific directory deletion upon account deactivation
- Authentication (Registration, login, and password changes).
- JSON-based user settings implementation for loading user session parameters (e.g. Dark Mode, and more, does not require changing database models since extra key-value pairs may simply be added on top of existing parameters).
- Self-configuring application through a custom script and loaded environment variables.
- Filename collision avoidance by UUID generation on uploaded files.
- Login viewer to redirect unauthorized users to the login page upon accessing restricted pages.
- Navbar changes depending on whether a user is logged in.
- Custom error handling pages, with some added easter eggs and humor sprinkled throughout.
- Responsive site layout using Bootstrap.
- Database creation script for rapid remote database creation.
- Test data generated through a test script that populates the site with data.
- File download from static directory.
- Dark Mode Vs. Light Mode, with icons to match.
- User profiles contain post counts comment counts, along with the account creation date displayed in a user-friendly format.
- Autoplay photo carousels on the About page.
- Comment and post deletion option available only to the user.
- Login requirement for any content deletion, preventing unintended actions from malicious users.
- Disabling buttons when an action is not possible (No avatar set – can't delete avatar)
- Flash messages for various errors and feedback.
- Using session data to track the current page for a smoother color mode toggling experience.

## Appendix B – All Pages

### The Portfolio – Main Page

**The Tech Portfolio**

**"Shopify Integration Gateway"**  
by [admin](#)

A Flask-based RESTful API that allows Shopify merchants to directly interface with Snoonu's own Fleet Management System, FalconFlex. Allowing for Shopify order fulfillments to be directly sent to the FalconFlex system through HMAC-secured webhooks. Granular fulfillment progress updates are then sent back to the merchant through the gateway through another set of webhooks.

### About Me

**About me**

I'm a person who dabbles in many skills in search for something fascinating. I've done bits in media production, entrepreneurship, administration, office work and programming (both low and high level). I have graduated with a Bachelor's of Science degree in Computer Engineering from HBUK. I have also studied at universities like Texas A&M University at Qatar and Carnegie Mellon University - Qatar.

I'm a fan of everything tech and always keep tabs on new stuff; semiconductor development, graphics technologies, audiophile gear etc. I also love video games, perfumery and music. I'm currently based in Wales, UK studying for a Master's in Computing and IT Management at Cardiff University.

## User Profile

The screenshot shows a web browser window displaying a user profile page. The browser's address bar shows the URL: `c22076452-cmt120-cw2-prod-c22076452-cmt120-cw2.apps.openshift.cs.cf.ac.uk/user/1`. The page header includes the blog name "Selman's Blog" and navigation links: Home, About, My Profile, New Post, Logout, and Settings. A "Download CV" button is visible in the top right corner. The profile section features a profile picture of a person with dark hair, the name "admin Profile", and the text "Member since 15.01.2023". Below the profile information, there is a section titled "Posts (6)". The first post is titled ["Shopify Integration Gateway"](#) and includes a thumbnail image of a hand-drawn diagram on a whiteboard. The diagram shows a complex system architecture with various components and connections. The post description reads: "A Flask-based RESTful API that allows Shopify merchants to directly interface with Snoonu's own Fleet Managemen...". At the bottom of the post, it states "Number of comments: 0".

## Post Creation Page

The screenshot shows a web browser window displaying a post creation page. The browser's address bar shows the URL: `c22076452-cmt120-cw2-prod-c22076452-cmt120-cw2.apps.openshift.cs.cf.ac.uk/create`. The page header includes the blog name "Selman's Blog" and navigation links: Home, About, My Profile, New Post, Logout, and Settings. A "Download CV" button is visible in the top right corner. The main heading is "Create a post". Below the heading, there are three input fields: "Title", "Content", and "Upload Picture". The "Upload Picture" field has a "Choose File" button and a "No file chosen" status. A "Submit" button is located at the bottom of the form. At the very bottom of the page, there is a copyright notice: "© 2023 - [selman.io](#) All rights reserved."

Selman's Blog CMT120 Assignm... +

Not secure | c22076452-cm1120-cw2-prod-c22076452-cm1120-cw2.apps.openshift.io

Google Facebook Outlook.com - sel... Inbox - selmantab... Instagram VirusTotal - Free On... PC Rep Steam Friend Reque... Qatar Address Look... ChaoTheSlayer #G... CSGO Exchange Hic... YT Music Other favorites

Selman's Blog Home About Login Sign Up Settings

Download CV

# Sign in

Username

Password

☐ Remember me

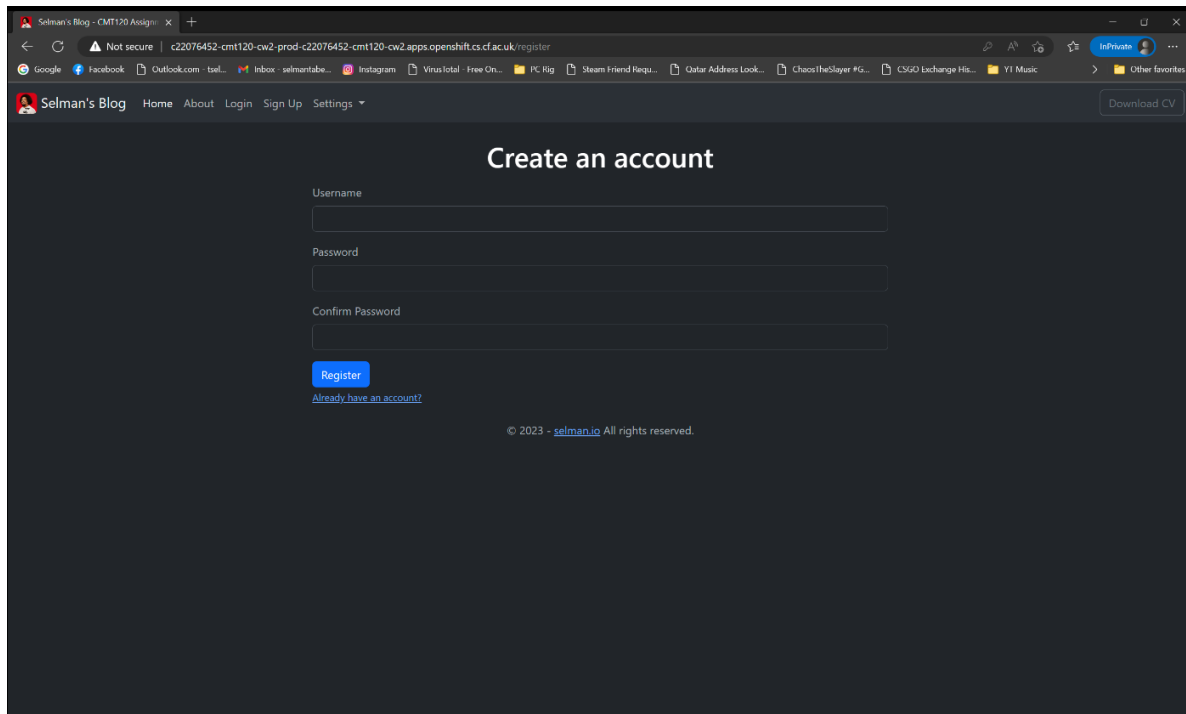
Login

[Don't have an account?](#)

© 2023 - [selman.io](#) All rights reserved.



## Signup Page



A screenshot of a web browser displaying the 'Create an account' page of 'Selman's Blog'. The browser's address bar shows the URL 'c22076452-cmt120-cw2-prod-c22076452-cmt120-cw2.apps.openshift.cs.cf.ac.uk/register'. The page has a dark theme and a navigation bar with links for Home, About, Login, Sign Up, and Settings. The main content area is titled 'Create an account' and contains three input fields for 'Username', 'Password', and 'Confirm Password'. Below these fields is a blue 'Register' button and a link for 'Already have an account?'. A copyright notice at the bottom reads '© 2023 - selman.io All rights reserved.'

Selman's Blog - CMT120 Assignment X

Not secure | c22076452-cmt120-cw2-prod-c22076452-cmt120-cw2.apps.openshift.cs.cf.ac.uk/register

Selman's Blog Home About Login Sign Up Settings

Download CV

### Create an account

Username

Password

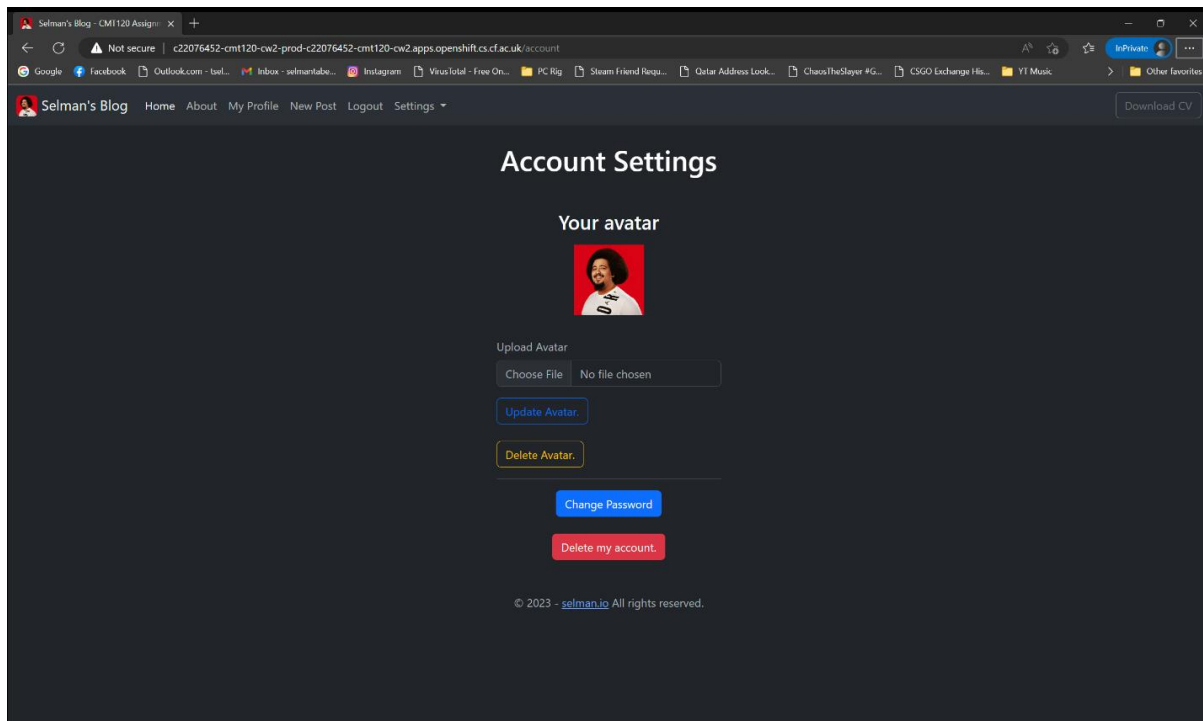
Confirm Password

Register

[Already have an account?](#)

© 2023 - [selman.io](#) All rights reserved.

## Account Settings



A screenshot of a web browser displaying the 'Account Settings' page of 'Selman's Blog'. The browser's address bar shows the URL 'c22076452-cmt120-cw2-prod-c22076452-cmt120-cw2.apps.openshift.cs.cf.ac.uk/account'. The page has a dark theme and a navigation bar with links for Home, About, My Profile, New Post, Logout, and Settings. The main content area is titled 'Account Settings' and features a section for 'Your avatar' with a profile picture. Below the picture are buttons for 'Upload Avatar' (with 'Choose File' and 'No file chosen' options), 'Update Avatar', and 'Delete Avatar'. Further down are buttons for 'Change Password' and 'Delete my account'. A copyright notice at the bottom reads '© 2023 - selman.io All rights reserved.'

Selman's Blog - CMT120 Assignment X

Not secure | c22076452-cmt120-cw2-prod-c22076452-cmt120-cw2.apps.openshift.cs.cf.ac.uk/account

Selman's Blog Home About My Profile New Post Logout Settings

Download CV

### Account Settings

Your avatar

Upload Avatar

Choose File No file chosen

Update Avatar

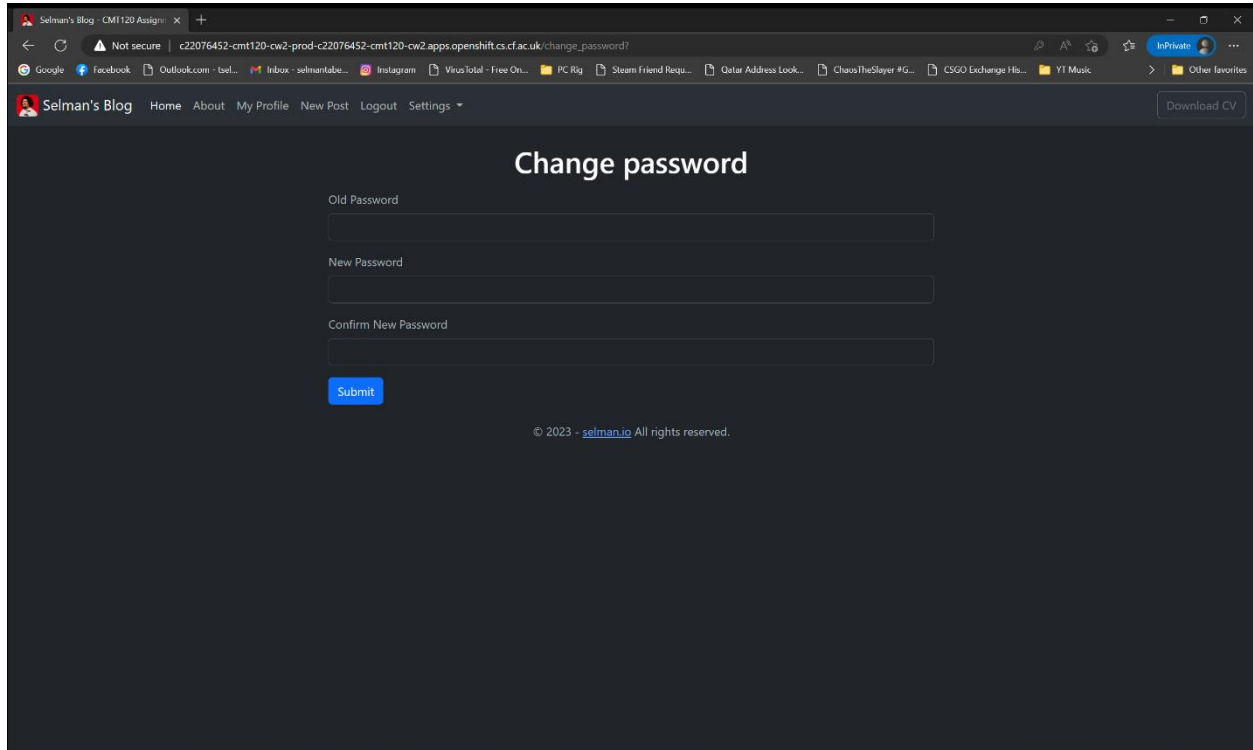
Delete Avatar

Change Password

Delete my account

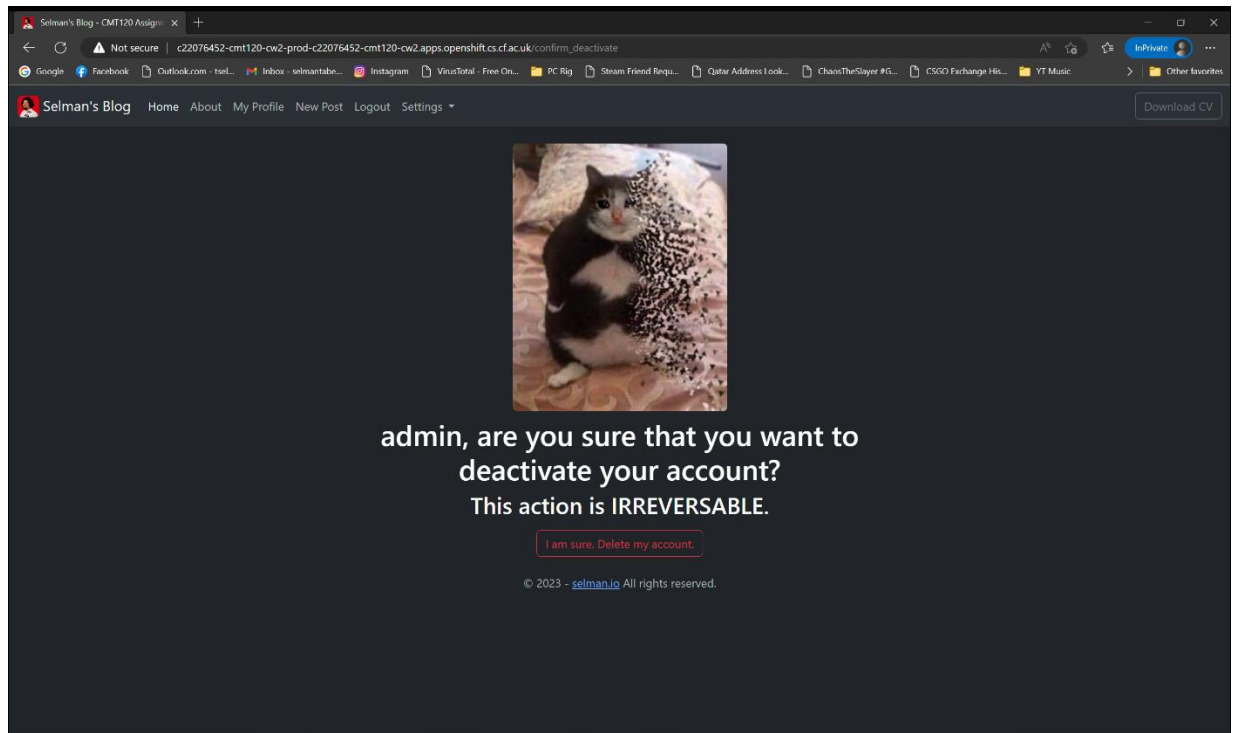
© 2023 - [selman.io](#) All rights reserved.

## Password Change Page



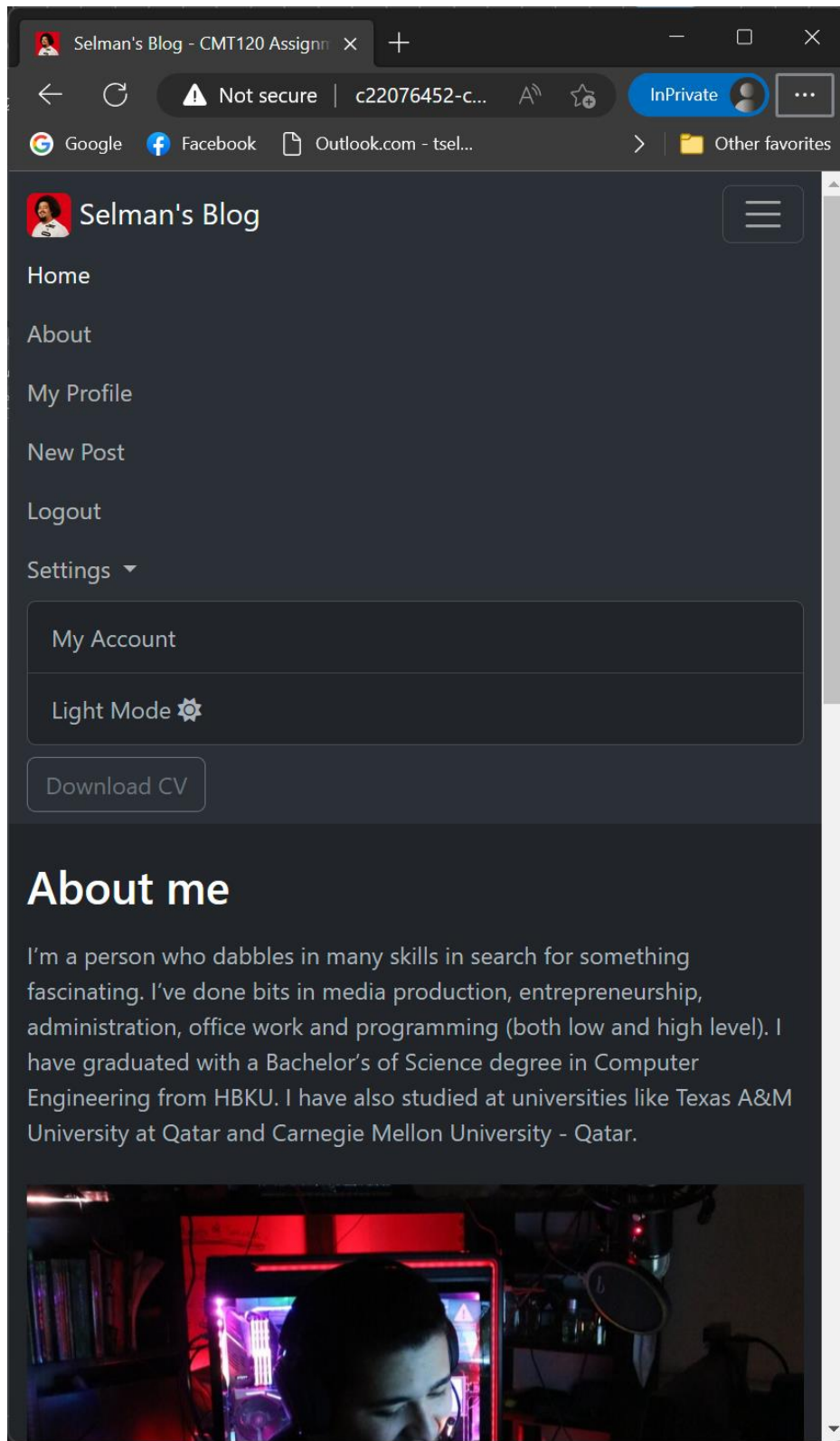
A screenshot of a web browser displaying a password change form. The browser's address bar shows the URL: `c22076452-cmt120-cw2-prod-c22076452-cmt120-cw2.apps.openshift.cs.cf.ac.uk/change_password/`. The page has a dark theme and a navigation bar at the top with links: Home, About, My Profile, New Post, Logout, and Settings. A 'Download CV' button is on the right. The main heading is 'Change password'. Below it are three input fields: 'Old Password', 'New Password', and 'Confirm New Password'. A blue 'Submit' button is at the bottom of the form. The footer text reads: '© 2023 - [selman.io](#) All rights reserved.'

## Account Deactivation Page

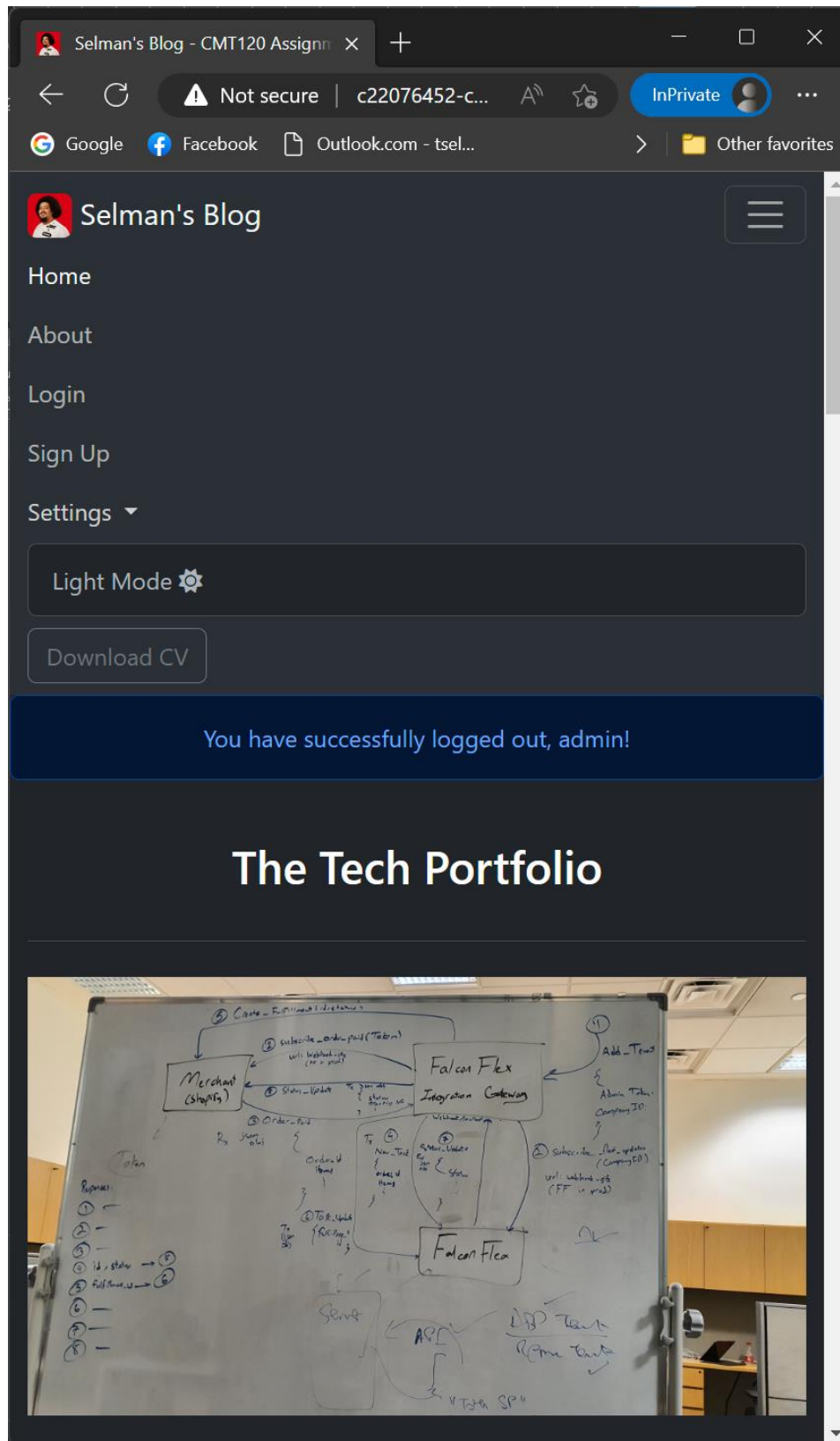


A screenshot of a web browser displaying an account deactivation confirmation page. The browser's address bar shows the URL: `c22076452-cmt120-cw2-prod-c22076452-cmt120-cw2.apps.openshift.cs.cf.ac.uk/confirm_deactivate`. The page has a dark theme and a navigation bar at the top with links: Home, About, My Profile, New Post, Logout, and Settings. A 'Download CV' button is on the right. The main content area features a large image of a black and white cat sitting on a patterned surface. Below the image, the text reads: 'admin, are you sure that you want to deactivate your account? This action is IRREVERSABLE.' A red button with the text 'I am sure. Delete my account.' is centered below the text. The footer text reads: '© 2023 - [selman.io](#) All rights reserved.'

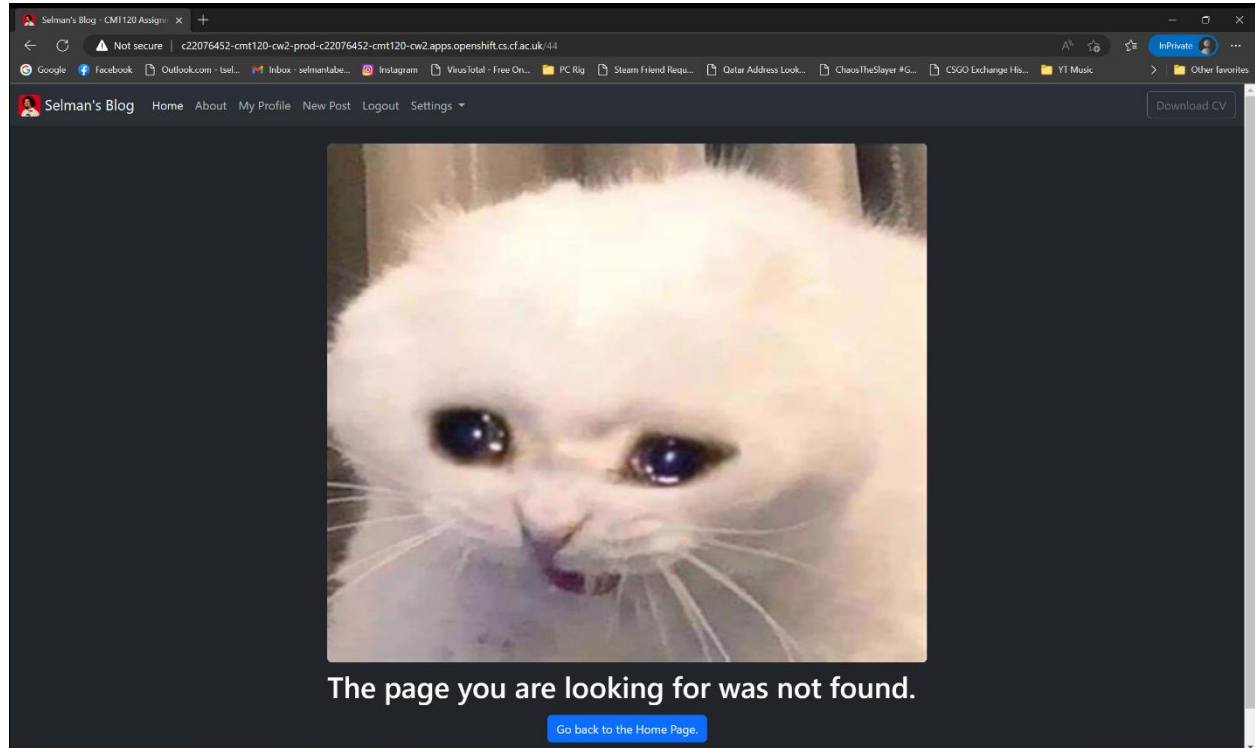
## Burger Popup Menu (Logged in)



## Burger Popup Menu (Logged out)



## Error 404 – Page Not Found



## Error 403 – Forbidden

