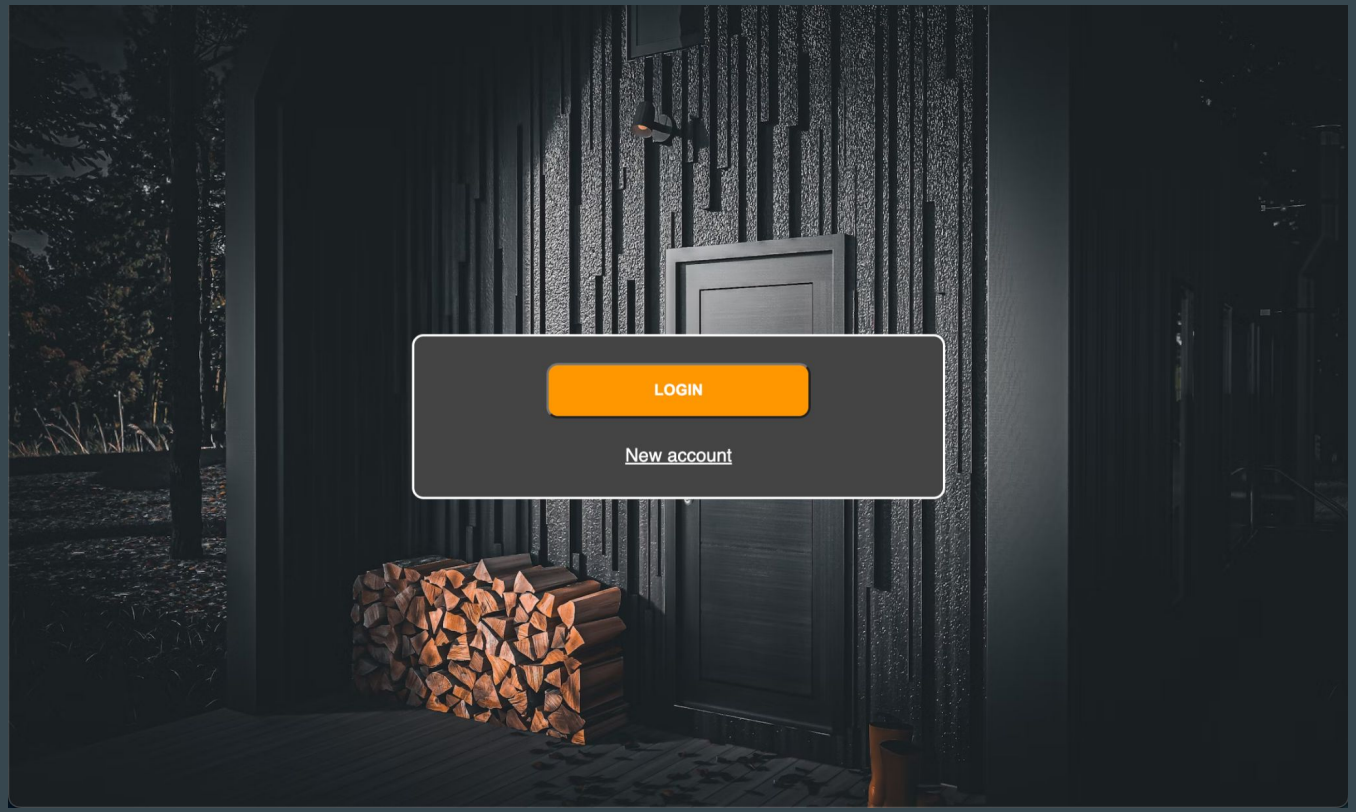
Scrambler Password

Khalfani Bozeman
Caleb Tsai
Danyal Aamir
James Pham

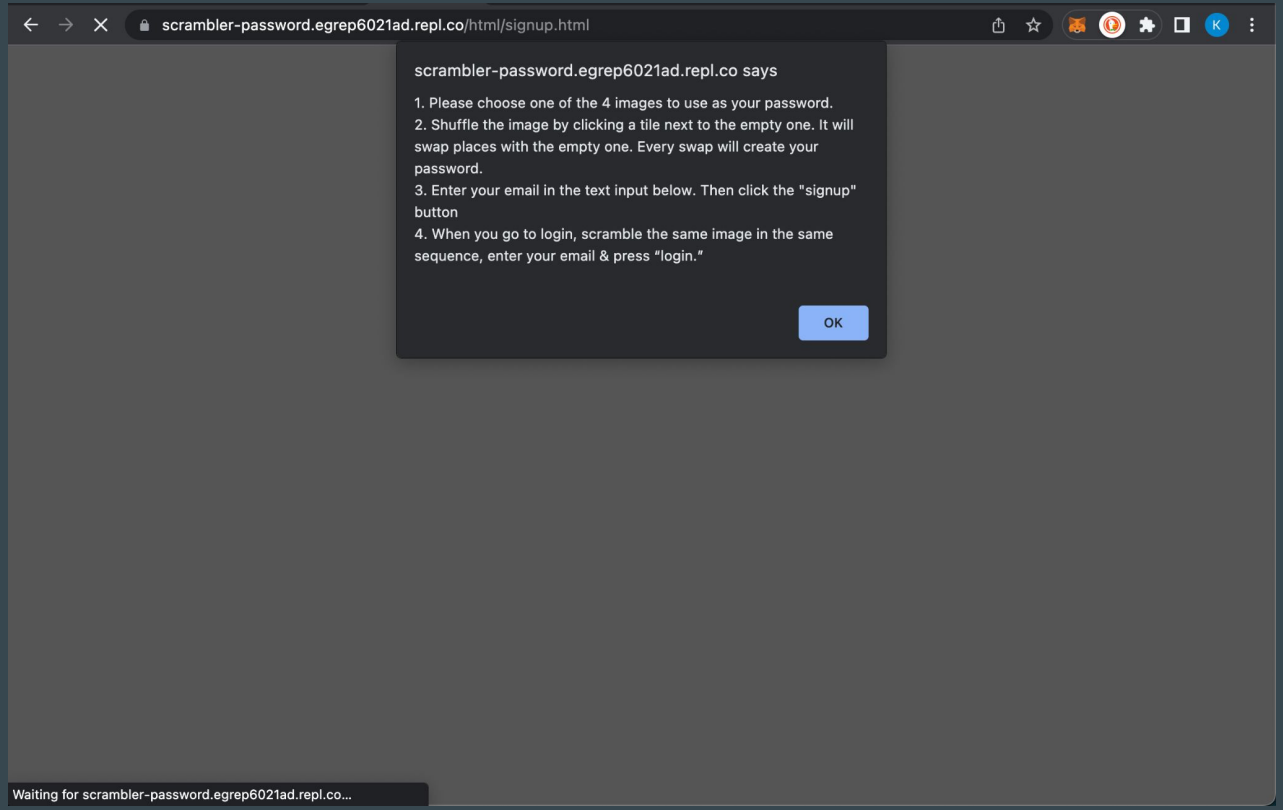
Demo
(https://scrambler-password.egrep6021ad.repl.co/)

Functional Requirements

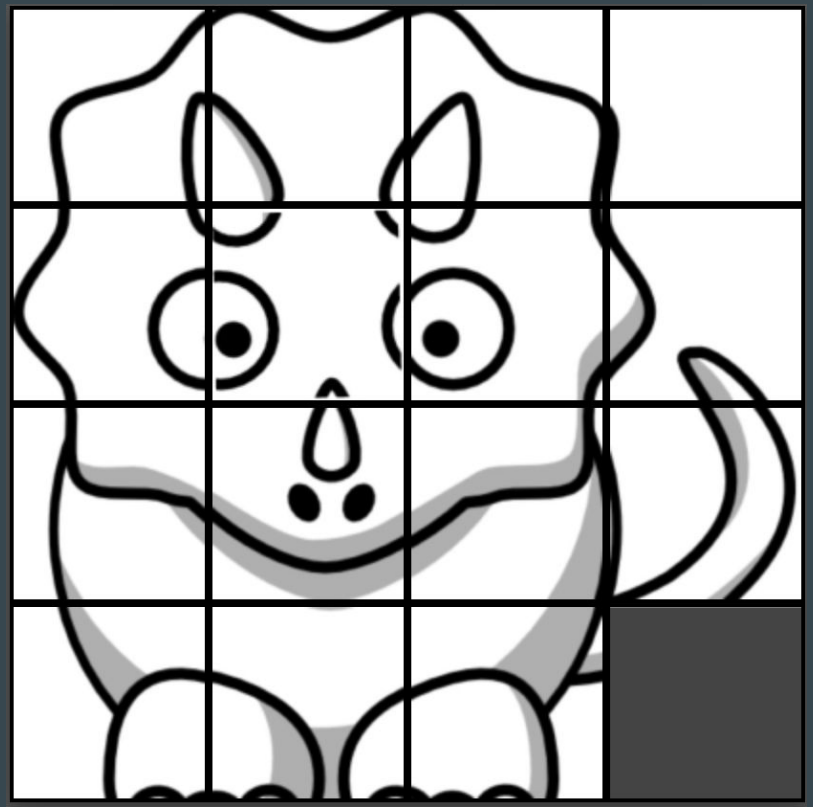
“The system shall allow a new user to signup & an existing user to login.”

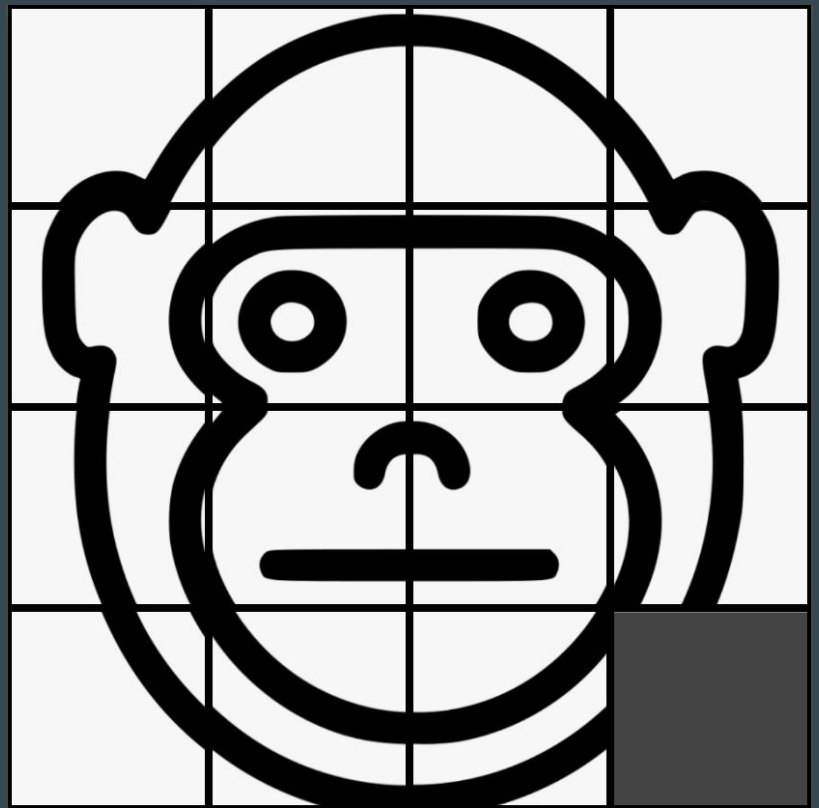


“The system shall allow a user to use their email as a username.”

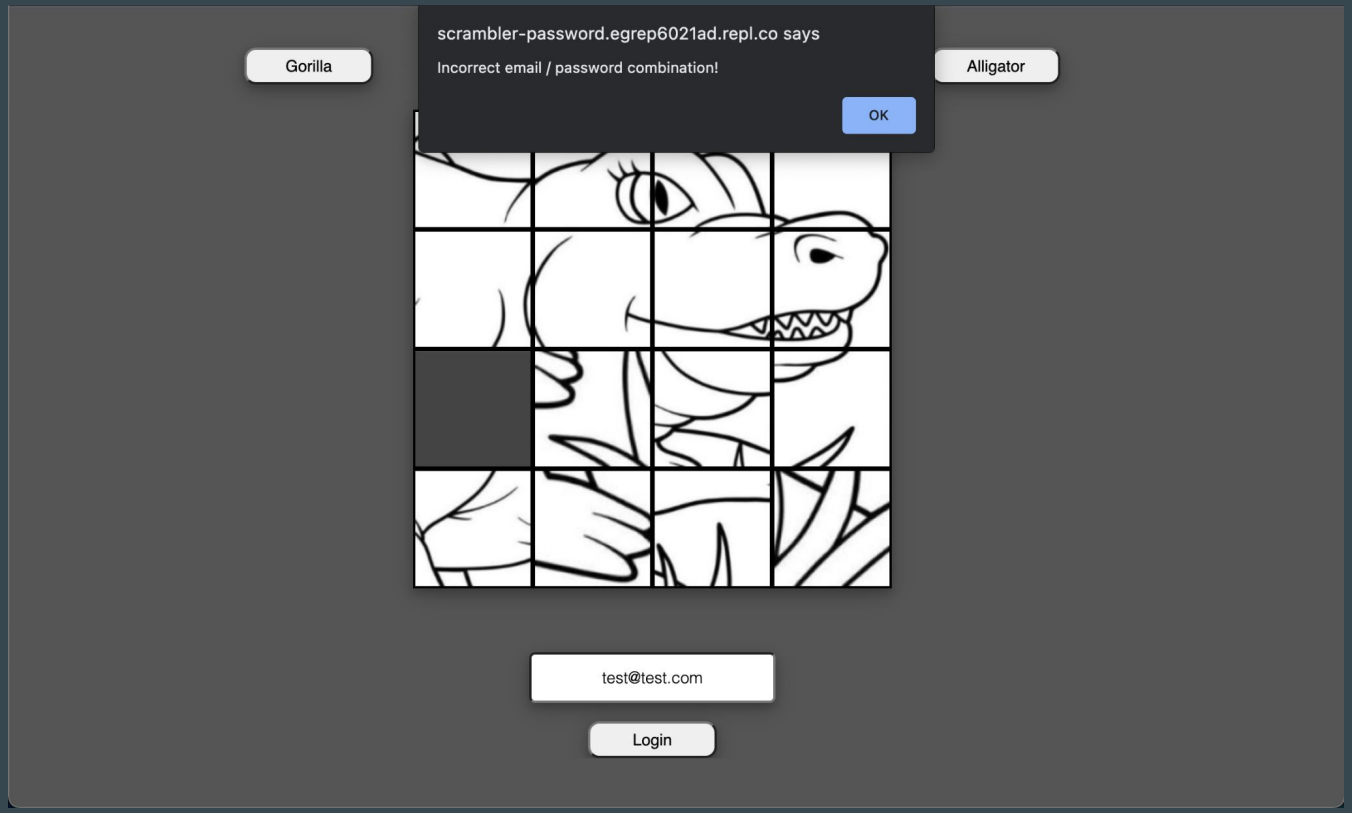


“The system shall allow a user to pick one image amongst the four choices.”

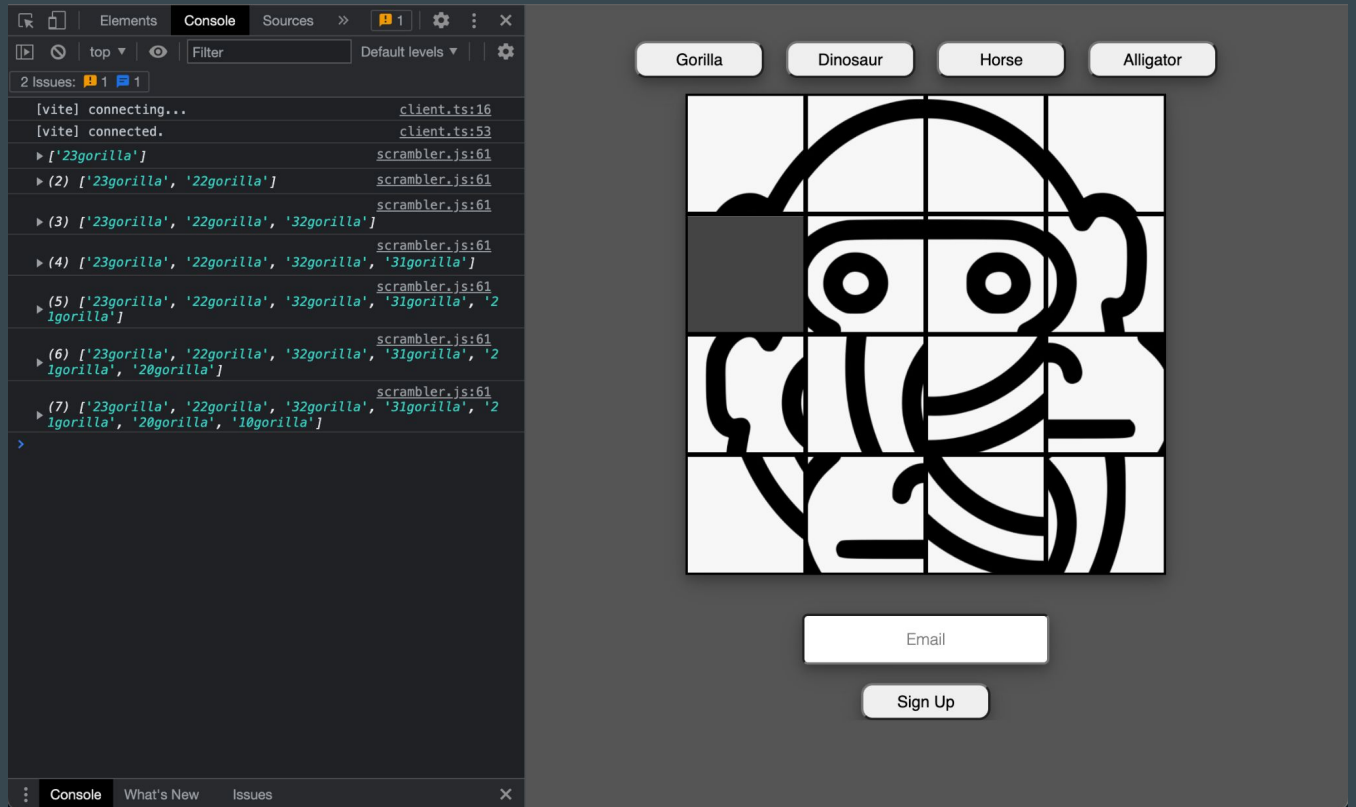




“The system shall notify users of any incorrect username / password combinations.”



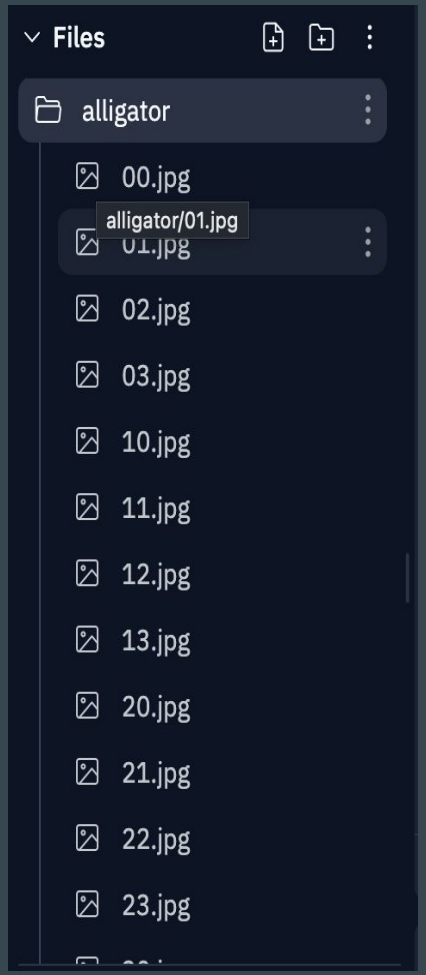
“The system shall allow a user to create a password by scrambling their choice of image.”



UI Logic

```
// Initialize the original table:
window.onload = () => {
  for (let i = 0; i <= 3; i++) {
    for (let j = 0; j <= 3; j++) {
      const elem = document.createElement('div');
      elem.style.backgroundImage = 'url("../' +
        startImage + '/' + i + '' + j + '.jpg")';
      elem.setAttribute('class', 'non_click_image');
      // Give each tile a unique ID
      elem.setAttribute('id', i + '' + j);
      // Add click handler to everybutton. Functions argument =
tile's ID
      elem.setAttribute('onClick', 'handleClick(' + i + ',' + j +
')');
      document.getElementById('row' + i).appendChild(elem);
    }
  }
```

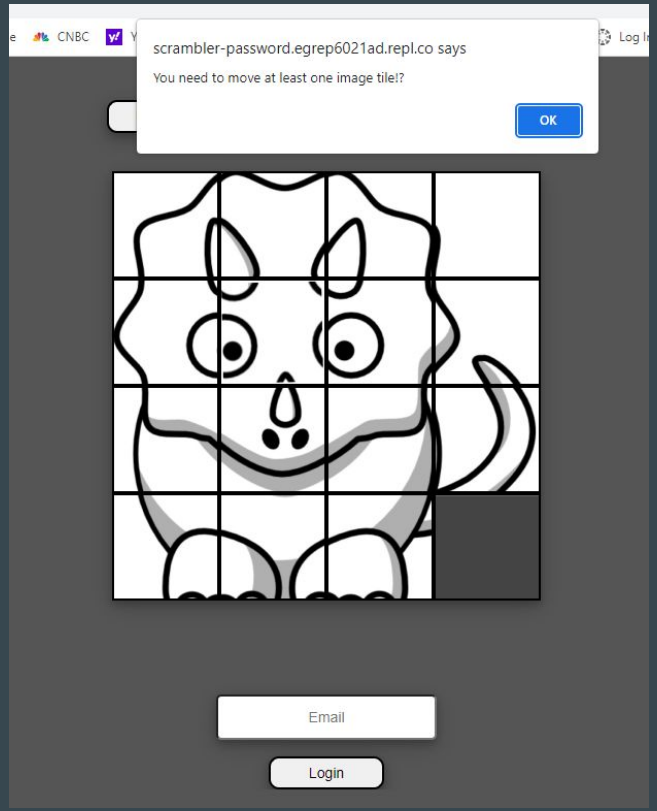
```
// Function to switch tiles / * track password * :
const handleClick = (row, col) => {
  const id = row + '' + col;
  const clickedImage = document.getElementById(id);
  if (clickedImage.getAttribute('class') == 'clickable_image') {
    swapPhotos(id);
    // Add the tile to password!
    password = [...password, id + chosenImage]
    console.log(password)
    // Re-assign "clickable_images"
    check()
  }
};
```




```
// Helper function to swap the tiles:
const swapPhotos = (id) => {
  const curr = document.getElementById(id);
  const currId = curr.id;
  const currSrc = curr.style.backgroundImage;
  const currClicker = curr.getAttribute('onClick');
  // Blank tiles ID is always '33':
  const blankTile = document.getElementById('33');
  // Set curr's attributes to blankTile's
  curr.style.backgroundImage = blankTile.style.backgroundImage;
  curr.id = blankTile.id;
  curr.setAttribute('onClick', blankTile.getAttribute('onClick'));
  // Set blankTile's attributes to to elem's originals
  blankTile.style.backgroundImage = currSrc;
  blankTile.id = currId;
  blankTile.setAttribute('onClick', currClicker);
};
```

Non-Functional Requirements

“The system prompts shall not accept blank usernames or passwords.”



“The system has a database that is always online and accessible through https protocols.”

```python
# Login API Enpoint:
@app.route('/login/', methods=('GET', 'POST'), strict_slashes=False)
def accept_password():
  print("[USER LOGIN::]")
  # Parse JSON data:
  data = request.get_json()
  print('USER INPUT:')
  print(f'\tUsername: {data["email"]}')
  print(f'\tPassword: {data["password"]}')
  # construct a string based on moves in array
  passwrd = ''.join([ele for ele in data['password']])
  # Validate user credentials:
  res = validate_password(data['email'], passwrd)
  # IF credentials are valid:
  if res:
    print('[USER VALIDATED::]')
    return jsonify(True)
  else:
    print('[USER CREDENTIALS WERE NOT VALIDATED::]')
    return jsonify(False)
```

"The system shall have the ability to implement CRUD operations on the database."

```
def add_user(user: str, passwrd: str, engine=engine):
  if check_existing_user(user, passwrd):
    return False

  # Creates fernet object to encrypt password
  key = os.environ['fernet_key']  # stored as a secret
  fernet = Fernet(key)
  encrypted = fernet.encrypt(passwrd.encode())

  statement = (  # SQL insert statement
    insert(User_Credentials).values(username=user, password=encrypted))

  engine.execute(statement)
  return True
```

“The system shall authenticate users given a username and array of moved tiles.”

```
172.31.128.1 - - [30/Nov/2022 01:59:59] "OPTIONS /register/ HTTP/1.1" 200 -
[USER REGISTRATION::]
USER INPUT:
    Username: 1234567@gmail.com
    Password: ['32dinosaur', '22dinosaur']
[USER REGISTERED::]
172.31.128.1 - - [30/Nov/2022 01:59:59] "POST /register/ HTTP/1.1" 200 -
```

“The system shall display a verification message when the user successfully logs in.”

Project Goals / Purpose

- The purpose of this project was to implement a web application that allows users to choose an alternative form of authentication, opposed to a traditional password. We did this through implementing a system that allows users to choose an image, and then scramble it in any sequence they see fit. The scrambled sequence is what they will use as their password for authentication purposes.

- Goals:
  - Demonstrate the ability to create a requirements sheet to act as an outline for the project.
  - Demonstrate the ability to use UML diagrams to represent the projects structure and working components.
  - Demonstrate the ability to work as a team to implement the projects essential features.
  - Demonstrate the ability to implement a REST API to use with HTTP protocols.
  - Demonstrate the ability to ensure no repetitive logins can be created.

Accomplishments

- Implemented functioning full-stack web application/
- Implemented a REST API using Python and the Flask web framework.
- Implemented a user interface using HTML, CSS and JavaScript.
- Implemented a database on the backend server using SQL.

Thank you!