E-VOTY - Tunisian Parliamentary election REST API

IT325 Web Services Final Project

by

Asma Aloui

January 2023

BA/IT Senior Student

Tunis Business School

Ben Arous, TUNISIA.

2022-2023

Declaration of Academic Ethics

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I declare that I have properly and accurately acknowledged all sources used in the production of this report.

I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: January 06th, 2023 Asma Aloui

Abstract

The 2022 Tunisian parliamentary elections are the fifteenth legislative elections in Tunisia and the fourth after the Tunisian Revolution, in which the Tunisian People's Assembly will be elected for its third parliamentary term, on December 17, 2022, and are overseen by The Independent Higher Authority for Elections (ISIE) [1].

The total percentage of voters was 8.8 % [2]. The age group over 60 years old ranked first in terms of voting, with a percentage of 34.8 %. The age group between 46 and 60 years old ranked second, with a percentage of 32.7 % and the age group between 26 and 45 years old represented 26.7 %. Only 5.8 % of those aged between 18 and 25 participated

We can detect a large churn rate in voting, especially between youth voters. This might be caused by the traditional voting process which is lengthy, time-consuming, and ambiguous.

On the other hand, the costs of preparing for elections in Tunisia are high, as it involves many logistical challenges. Some of the expenses that may be involved in preparing for elections in Tunisia include:

-Printing and distributing ballot papers and voter information materials

-Hiring and training staff to work at polling stations

-Renting or building polling stations

-Transporting polling materials and staff to polling stations

-Setting up voting booths and other necessary equipment

-Ensuring the security of polling stations and materials

-Conducting voter education and outreach programs.

To increase the efficiency of elections regarding costs, accuracy, transparency, and participation, we should opt for digitizing elections.

Keywords - Digitisation, Elections, Tunisia.

Contents

Chapter 1

Overview

1.1 Introduction

My final project for the IT325 Web Services course this semester consists of a REST-ful API for parliamentary elections that utilizes the Flask framework and various libraries for database management, and user authentication/authorization. The API supports token-based authentication with Flask-JWT-Extented with a "roles" feature that grants each user a set of privileges according to their role (admin or user), SQLAlchemy for database management [3] and Marshmallow for serialization and deserialization of objects [4].

1.2 Objective

The purpose of this web service is to digitize the election process without losing its integrity and security. Voters create their digital profiles: username and password and fill in the needed information such as their ID, first name and last name , governate, and electoral circle. On election day, they log in and vote on one candidate that exists in their respective electoral circle(Only one vote is counted). A verification email is sent after voting.
This system also has an admin that is responsible for creating the candidate profiles: ID, name, governate, and electoral circle.

1.3 Motivation

The Motivation behind this project is to sustain this democratic procedure without having heavy costs. As elections generally have the same structure and occur repetitively, it is inefficient to allocate pricey budgets for each election. So, the project itself is very scalable to cover Presidential elections, Municipal elections, Referendums, Elections for regional and professional councils, and more. Thus, E-Voty's main motivation is to make the voting process as easy, fast, and accessible for every citizen.

Chapter 2

Authentication and Security

2.1 Registration and roles

2.1.1 Registration

To access the API as a user, an account must be created. Each user is assigned a unique ID. Accounts can be created by visiting the "localhost/user" endpoint and providing a ID, username, email, password, first name, last name, governate, electoral circle, and role as follows:

2.1.2 Roles

There are two types of users: administrators and normal users. Each type has different privileges based on their roles, as shown in the accompanying figure:

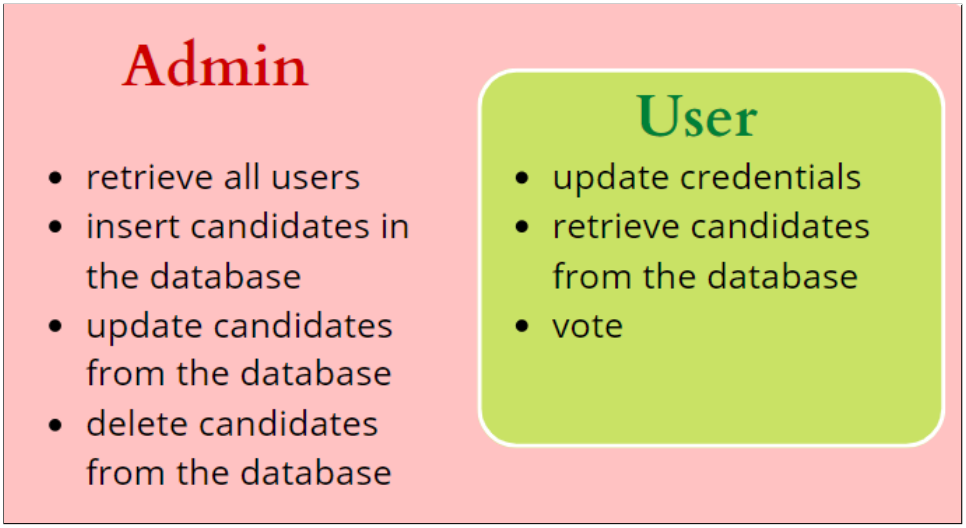

Figure 2.1: Roles hierarchy and privileges..

2.2 Security

After successfully completing the registration process, users must log in to the system and use the token provided to them in order to gain access to the voting feature. This token acts as a form of verification, ensuring that only authorized users are able to participate in the voting process. Additionally, this token-based system provides an extra layer of security, as it prevents unauthorized access to the voting feature and helps to safeguard the integrity of the voting process.

2.2.1 Password

I used the "pbkdf2-sha256" function which is a password hashing function that uses the PBKDF2 (Password-Based Key Derivation Function 2) algorithm with the SHA-256 hash function to generate a secure hash of a password to store password hashes in a secure way so that the plaintext password cannot be easily retrieved from the hash.

2.2.2 Login

Users must visit the "localhost/login" endpoint and enter their username, email, and password to log in. Upon successfully submitting these credentials, the API will return a response containing the user's details.

```
{
  "access_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE2NzI0NTYzMzQsImlhd
CI6MTY3MjQ1MjczNCwibmJmIjoxNjcyNDUyNzM0LCJqdGkiOiIxMWVkNDgyZi02OGZj
LTQxNmUtODIyYi0wMTM3N2ZlMmYxNTMiLCJpZGVudGl0eSI6MTE0MjgxMDAsImZyZXN
oIjpmYWxzZSwidHlwZSI6ImFjY2VzcyIsInVzZXJfY2xhaW1zIjp7fX0.3WR1Q7vLq5
TruktBHuJY-Yc9xayyHk1zcsXgUj7B1Kg"
}
```


2.2.3 Token

Each application requires a JSON Web Token (JWT) [5] access token for authentication. The token includes information about the user, including their role, which allows for the implementation of user-specific privileges. The process of JWT Authentication is illustrated in the following diagram:

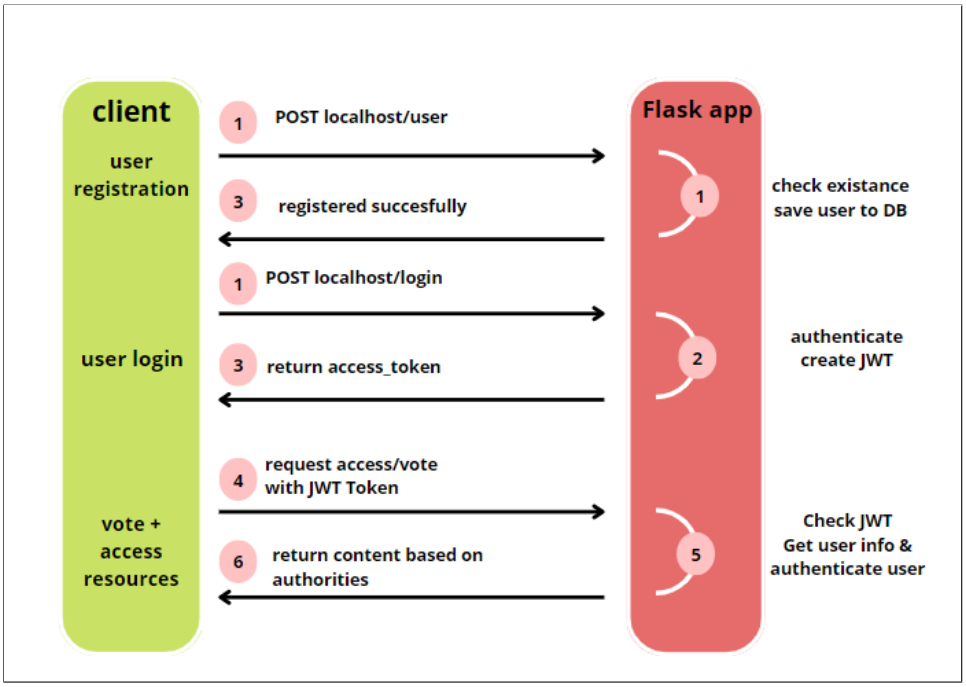

Figure 2.2: Signup Login with JWT Authentication.

Chapter 3

HTTP Methods and Code

3.1 User Requests

3.1.1 GET requests

GET /candidate

Get the candidate's name, governate, and electoral circle of all candidates in Tunisia.

```python
# Get all candidates
@app.route('/candidate', methods=['GET'])
def get_candidates():
    all_candidates = Candidate.query.all()
    result = candidates_schema.dump(all_candidates)
    return jsonify(result)
```

GET /candidate/<electoral-circle>

Get the list of candidates of the same chosen electoral circle.

```python
# Get candidates by electoral circle
@app.route('/candidate/<electoral_circle>', methods=['GET'])
def get_candidate(electoral_circle):
    candidates = Candidate.query.filter_by(electoral_circle=electoral_circle).all()
    result = candidates_schema.dump(candidates)
    return jsonify(result)
```

3.1.2 POST requests

POST /vote

a POST request to this route with a valid JWT will allow the user to vote for a candidate. The user's 'voted' field will be updated to True and a confirmation email will be sent to the user's email address.

```python
# Confirm identity and vote
@app.route('/vote', methods=['POST'])
@jwt_required
def vote():
    user_id = request.json['user_id']
    candidate_name = request.json['candidate_name']

    user = User.query.get(user_id)
    if user.voted:
        return jsonify({'message': 'You have already voted!'})

    candidate = Candidate.query.filter_by(name=candidate_name).first()
    if candidate is None:
        return jsonify({'message': 'Candidate not found!'})
    if candidate.governate != user.governate or candidate.electoral_circle != user.electoral_circle:
        return jsonify({'message': 'Invalid candidate for your governate and electoral circle!'})

    user.voted = True
    db.session.commit()

    # Send email confirmation
    send_confirmation_email(user.email, candidate_name)

    return jsonify({'message': 'Vote successful!'})
```

```
{
    "message": "Vote successful!"
}
```

3.1.3 PUT requests

PUT /user/<id>

When this route is called, it first gets the current user's ID from the JSON Web Token (JWT) in the request header and queries the User model to get the current user. If the current user's role is not 'user', it returns an HTTP 401 Unauthorized response. Otherwise, it gets the user with the specified ID from the database and updates the user's fields with the values provided in the

request body. The password is hashed before it is stored in the database. Finally, it commits the changes to the database and returns the updated user's information in the response.

```python
# Update a user
@app.route('/user/<id>', methods=['PUT'])
@jwt_required
def update_user(id):
    # Get the current user's ID
    current_user_id = get_jwt_identity()

    # Query the User model to get the current user
    current_user = User.query.filter_by(id=current_user_id).first()
    if current_user.role != 'user':
        return jsonify({'message': 'Unauthorized'}), 401

    user = User.query.get(id)
    username = request.json['username']
    password = request.json['password']
    email = request.json['email']
    first_name = request.json['first_name']
    last_name = request.json['last_name']
    governate = request.json['governate']
    electoral_circle = request.json['electoral_circle']
    role = request.json['role']
    hashed_password = pbkdf2_sha256.hash(password)
    user.username = username
    user.password = hashed_password
    user.email = email
    user.first_name = first_name
    user.last_name= last_name
    user.governate = governate
    user.electoral_circle = electoral_circle
    user.role = role
    db.session.commit()
    return user_schema.jsonify(user)
```

3.2 Admin Requests

3.2.1 GET requests

GET /user

this request gets the current user's ID from the JSON Web Token (JWT) in the request header and queries the User model to get the current user. If the current user's role is not 'admin', it returns an HTTP 401 Unauthorized response. If the current user is an admin, it retrieves all

users from the database using the User model's 'query.all()' method.

```python
# Get all users
@app.route('/user', methods=['GET'])
@jwt_required
def get_users():
    # Get the current user's ID
    current_user_id = get_jwt_identity()

    # Query the User model to get the current user
    current_user = User.query.filter_by(id=current_user_id).first()

    # Check if the current user is an admin
    if current_user.role != 'admin':
        return jsonify({'message': 'Unauthorized'}), 401

    # Get all users
    all_users = User.query.all()
    result = users_schema.dump(all_users)
    return jsonify(result)
```

GET /user/<id>

this request retrieves a singer user based on its ID . If the current user's role is not 'admin', it returns an HTTP 401 Unauthorized response.

```python
# Get a single user
@app.route('/user/<id>', methods=['GET'])
@jwt_required
def get_user(id):
    # Get the current user's ID
    current_user_id = get_jwt_identity()

    # Query the User model to get the current user
    current_user = User.query.filter_by(id=current_user_id).first()

    if current_user.role != 'admin':
        return jsonify({'message': 'Unauthorized'}), 401

    user = User.query.get(id)
    return user_schema.jsonify(user)
```

GET /candidate

Get the candidate's name, governate, and electoral circle of all candidates in Tunisia.

```python
# Get all candidates
@app.route('/candidate', methods=['GET'])
def get_candidates():
    all_candidates = Candidate.query.all()
    result = candidates_schema.dump(all_candidates)
    return jsonify(result)
```

3.2.2 POST requests

POST /candidate

a POST request to this route with a valid JWT will allow the admin to add a new candidate.

```python
# Create a candidate
@app.route('/candidate', methods=['POST'])
@jwt_required
def create_candidate():
    # Get the current user's ID
    current_user_id = get_jwt_identity()

    # Query the User model to get the current user
    current_user = User.query.filter_by(id=current_user_id).first()
    if current_user.role != 'admin':
        return jsonify({'message': 'Unauthorized'}), 401

    name = request.json['name']
    governate = request.json['governate']
    electoral_circle = request.json['electoral_circle']

    new_candidate = Candidate(name=name, governate=governate, electoral_circle=electoral_circle)
    db.session.add(new_candidate)
    db.session.commit()
    return candidate_schema.jsonify(new_candidate)
```

3.2.3 PUT requests

PUT /candidate/<id>

When this route is called, it retrieves the candidate with the specified ID from the database using the Candidate model's 'query.get(id)' method and updates the candidate's fields with the values provided in the request body. Finally, it commits the changes to the database and returns the updated candidate's information in the response, serialized using the candidate-schema object.

```python
# Update a candidate
@app.route('/candidate/<id>', methods=['PUT'])
@jwt_required
def update_candidate(id):
    current_user_id = get_jwt_identity()
    current_user = User.query.get(current_user_id)
    if current_user.role != 'admin':
        return jsonify({'message': 'Unauthorized'}), 401

    candidate = Candidate.query.get(id)
    name = request.json['name']
    governate = request.json['governate']
    electoral_circle = request.json['electoral_circle']

    candidate.name = name
    candidate.governate = governate
    candidate.electoral_circle = electoral_circle

    db.session.commit()
    return candidate_schema.jsonify(candidate)
```

3.2.4 Delete requests

Delete /candidate/<id>

When this route is called, it deletes the candidate with the specified ID from the database using the Candidate model's 'query.get(id)' method .

Delete /user/<id>

When this route is called, it deletes the user with the specified ID from the database using the user model's "query.get(id)" method .

```python
# Delete a candidate
@app.route('/candidate/<candidate_id>', methods=['DELETE'])
@jwt_required
def delete_candidate(candidate_id):
    # Get the current user object from the database
    current_user_id = get_jwt_identity()
    current_user = User.query.get(current_user_id)
    if current_user.role != 'admin':
        return jsonify({'message': 'Unauthorized'}), 401

    candidate = Candidate.query.get(candidate_id)
    db.session.delete(candidate)
    db.session.commit()
    return jsonify({'message': 'Candidate Deleted Succesfully'}), 200
```

```json
{
    "message": "Candidate Deleted Succesfully"
}
```

```python
# Delete a user
@app.route('/user/<id>', methods=['DELETE'])
@jwt_required
def delete_user(id):
    # Get the current user's ID
    current_user_id = get_jwt_identity()

    # Query the User model to get the current user
    current_user = User.query.filter_by(id=current_user_id).first()
    if current_user.role != 'admin':
        return jsonify({'message': 'Unauthorized'}), 401

    user = User.query.get(id)
    db.session.delete(user)
    db.session.commit()
    return jsonify({'message': 'User Deleted Succesfully'}), 200
```

```json
{
    "message": "User Deleted Succesfully"
}
```

3.3 Database Structure

Source of my database: isie.Tn .

The database consists of 2 tables with no relations between them.

3.3.1 Tables

Table "candidates"

containing 3 columns :

1. name: Name of the candidate of type text
2. governate: Name of the designated governate of type text
3. electoral circle: Name of the designated electoral circle of type text

Table "users"

containing 9 columns :

1. id : id of the user of type number
2. username : username of the user of type text
3. password : password of the user of type text
4. email : email of the user of type text
5. first name : first name of the user of type text
6. last name : Last name of the user of type text
7. governate : Governate of the user of type text
8. electoral circle : electoral circle of the user of type text
9. role : role of the user (User or Admin)

3.4 Insomnia Contribution

Insomnia is an open-source desktop application that takes the pain out of interacting with and designing, debugging, and testing APIs.
Insomnia combines an easy-to-use interface with advanced functionality like authentication helpers, code generation, and environment variables [6].

I used Insomnia to conduct automated testing of my API requests. Insomnia allows for the creation of test scripts, which can include various assertion statements such as " 200 OK", to ensure that the API is functioning as intended. This not only allows for more efficient testing, but also provides a greater level of confidence in the reliability of the API. Additionally, the use of Insomnia allowed me to easily troubleshoot any issues that may have arisen during the testing process and make any necessary adjustments to the API.

3.5 GIT contribution

In the terminal of the project, the following code is executed:

git init

git add.

git commit m " first commit"

git branch M main

git remote add origin github.com/asmaaloui/e-voty.git

git push u origin main

Chapter 4

Conclusion

The idea of this project came up to me on election day when I heard the news about the low rate of participation in Tunisia, especially among youth. After some discussions with my peers about this, we realized the depth of this problem on many dimensions: costs, accuracy, transparency, and participation. From these observations, I thought of why not digitise this process and make it accessible from the comfort of our homes, without having painful expenses each period. After some research about if this model was used before in other countries, I found out that it is very popular in advanced ones, like Canada and The USA [7]. My strong desire to improve our country's future and simplify our lives through the use of advanced technology is evident in my work, and I hope that this is conveyed clearly.

I am delighted to have had the opportunity to work on this project. Despite encountering more challenges than expected, I am grateful for the valuable lessons and personal growth I gained through this experience.

I would like to express my gratitude to Dr. Montassar Ben Messaoud, our professor, for his guidance and contribution in leading us to this point in our learning and challenging us constantly to surpass our boundaries.

Asma Aloui

TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

Bibliography

[1] “Tunisian parlimentary elections 2022.” ://www.isie.tn/. [Online; accessed Jan-2023].

[2] “Tunisie : participation très faible aux législatives boycottées par l’opposition.” https://www.france24.com/fr/afrique/20221217-tunisie-participation-tr

[3] “Sqlalchemy:the python sql toolkit and object relational mapper.” ://https://www.sqlalchemy.org/. [Online; accessed Jan-2023].

[4] “marshmallow: simplified object serialization.” ://marshmallow.readthedocs.io/en/stable/. [Online; accessed Jan-2023].

[5] “Json web tokens.” ://jwt.io/. [Online; accessed Jan-2023].

[6] “Introduction to insomnia.” ://docs.insomnia.rest/insomnia/get-started/. [Online; accessed Jan-2023].

[7] “Is e-voting currently used in any elections with emb participation?.” ://www.idea.int/data-tools/question-view/742. [Online; accessed Jan-2023].