Technical Report: Election Management System

Table of Contents

1 Introduction

1.1 Purpose

The purpose of this technical report is to document the design and implementation of the Election Management System. This system aims to provide a robust backend solution for managing user authentication and elections. The report outlines the goals, functionality, and key features of the system, offering insights into the technologies used and the application's overall structure.

1.2 Scope

The scope of this report encompasses the various components and functionalities of the Election Management System. It delves into the technologies employed, the organization of the application, and the workflow involved in user signup, authentication, and election management. The report concludes with an overview of achievements and potential areas for future improvements.

2 Technologies Used

2.1 Node.js

Node.js is a server-side JavaScript runtime that enables the execution of JavaScript code outside a web browser. It is widely used for building scalable and high-performance network applications.

2.2 Express.js

Express.js is a web application framework for Node.js, designed to simplify the process of building robust and scalable web applications and APIs.

2.3 MongoDB

MongoDB is a NoSQL database that stores data in flexible, JSON-like documents. It provides high performance, scalability, and flexibility for managing diverse data types.

2.4 Postman

Postman is a popular API testing tool that simplifies the process of developing, testing, and documenting APIs. It is used for testing the endpoints of the Election Management System.

2.5 JWT (JSON Web Token)

JSON Web Tokens are a compact, URL-safe means of representing claims between two parties. In the context of the Election Management System, JWTs are used for secure user authentication.

3 Application Structure

3.1 Controllers

Controllers serve as the bridge between the user interface and the application logic. In the context of the Election Management System, two main controllers are employed: `authController` and `electionController`. These controllers handle user authentication and election-related operations, respectively.

3.1.1 `authController`

The `authController` manages user authentication processes, including user signup and login. It interacts with the User Model to store and retrieve user data securely.

3.1.2 `electionController`

The `electionController` oversees operations related to elections, such as creating, updating, retrieving, and deleting election data. It interfaces with the Elections Model to interact with the MongoDB database.

3.2 Models

Models represent the structure and schema of data within the application. In this system, two models are defined:

3.2.1 `User Model`

The `User Model` defines the structure of user data, including fields such as name, email, password, and connection status. This model is crucial for managing user information during authentication.

3.2.2 `Elections Model`

The `Elections Model` specifies the structure of election data, including fields like title, description, and date. It enables consistent interaction with election-related information in the MongoDB database.

3.3 Routes

Routes define the endpoints through which the application's functionality is accessed. Two main route files are utilized:

3.3.1 `auth.route`

The `auth.route` file outlines the API endpoints related to user authentication, such as signup and login.

3.3.2 `election.route`

The `election.route` file manages API endpoints associated with election operations, including creation, retrieval, update, and deletion.

3.4 Middleware

Middleware functions enhance the functionality of the application by performing tasks between the request and response. The `checkAuth` middleware, in particular, is responsible for verifying user authentication using JSON Web Tokens (JWT).

3.5 App Configuration

App configuration settings are crucial for ensuring the smooth operation of the application. The Express app is configured to use necessary middleware, including CORS for cross-origin resource sharing.

3.5.1 Express App Configuration

Express app configuration sets up middleware, routing, and other essential settings to establish the groundwork for the Election Management System.

3.6 Server Setup

The server setup involves creating an HTTP server to handle incoming requests. The server listens on a specified port, allowing the application to be accessed through web browsers.

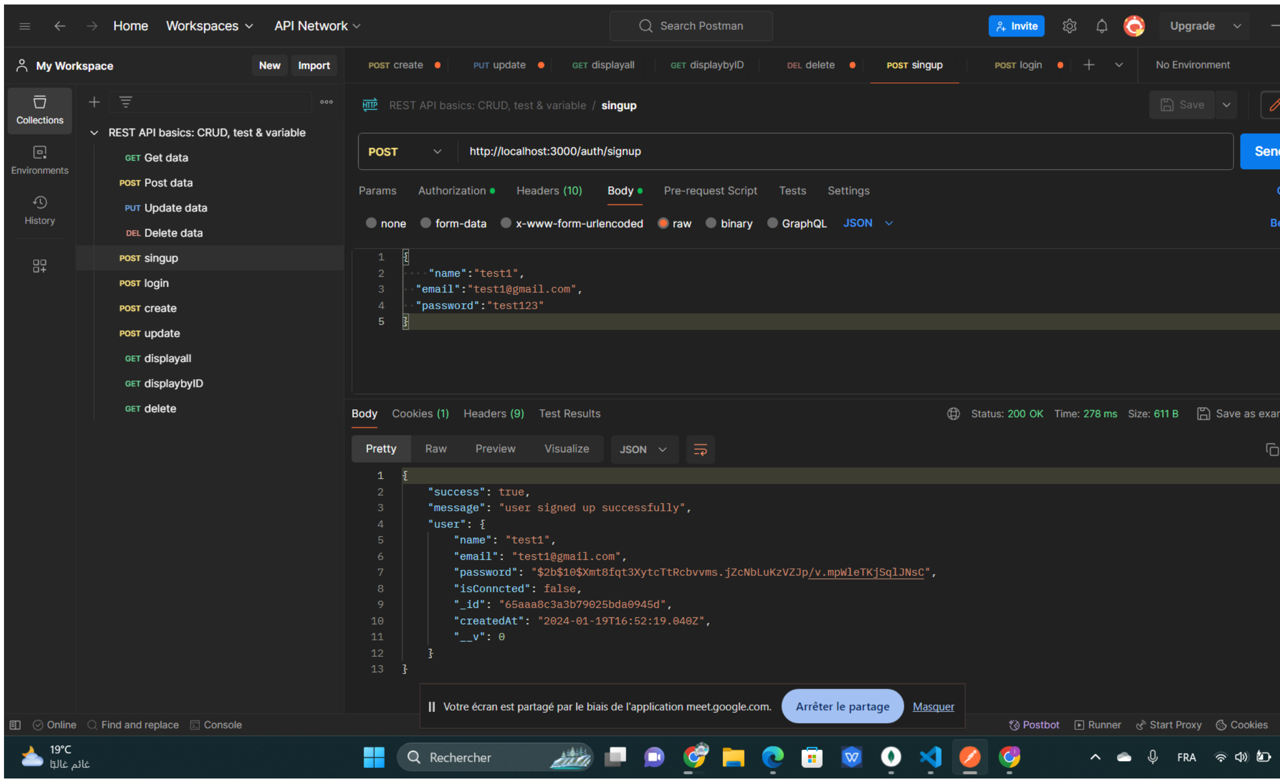
3.6.1 HTTP Server

The HTTP server is created using the `http` module in Node.js. It listens on a specified port, providing the necessary infrastructure for the Election Management System.

4 Workflow

4.1 User Signup

Endpoint: POST /auth/signup
Description: Allows users to register by providing a name, email, and password.
Actions: Validates user input.
Checks for existing email addresses.
Hashes the user's password using bcrypt.
Saves user data to the MongoDB database.



POST http://localhost:3000/auth/signup

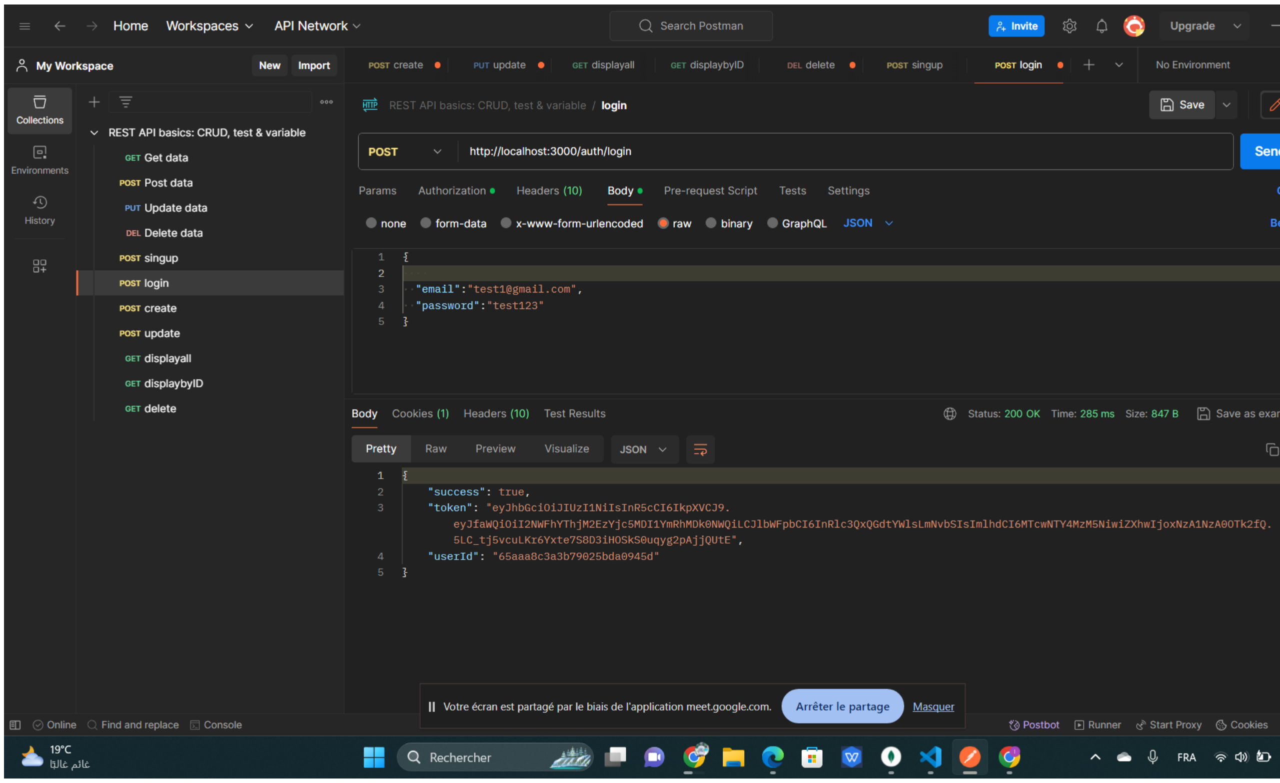
```
{
    "name":"test1",
  "email":"test1@gmail.com",
  "password":"test123"
}
```

Status: 200 OK Time: 278 ms Size: 611 B

```
{
    "success": true,
    "message": "user signed up successfully",
    "user": {
        "name": "test1",
        "email": "test1@gmail.com",
        "password": "$2b$10$Xmt8fqt3XytcTtRcbvvms.jZcNbLuKzVZJp/v.mpWleTKjSqlJNsC",
        "isConncted": false,
        "_id": "65aaa8c3a3b79025bda0945d",
        "createdAt": "2024-01-19T16:52:19.040Z",
        "__v": 0
    }
}
```

4.2 User Login

Endpoint: POST /auth/login
Description: Enables user login, issuing a JWT token upon successful authentication.
Actions: Verifies user credentials.
Compares hashed password with stored hash.
Generates a JWT token with user information.
Sends the token in the response.



REST API basics: CRUD, test & variable / login

POST http://localhost:3000/auth/login

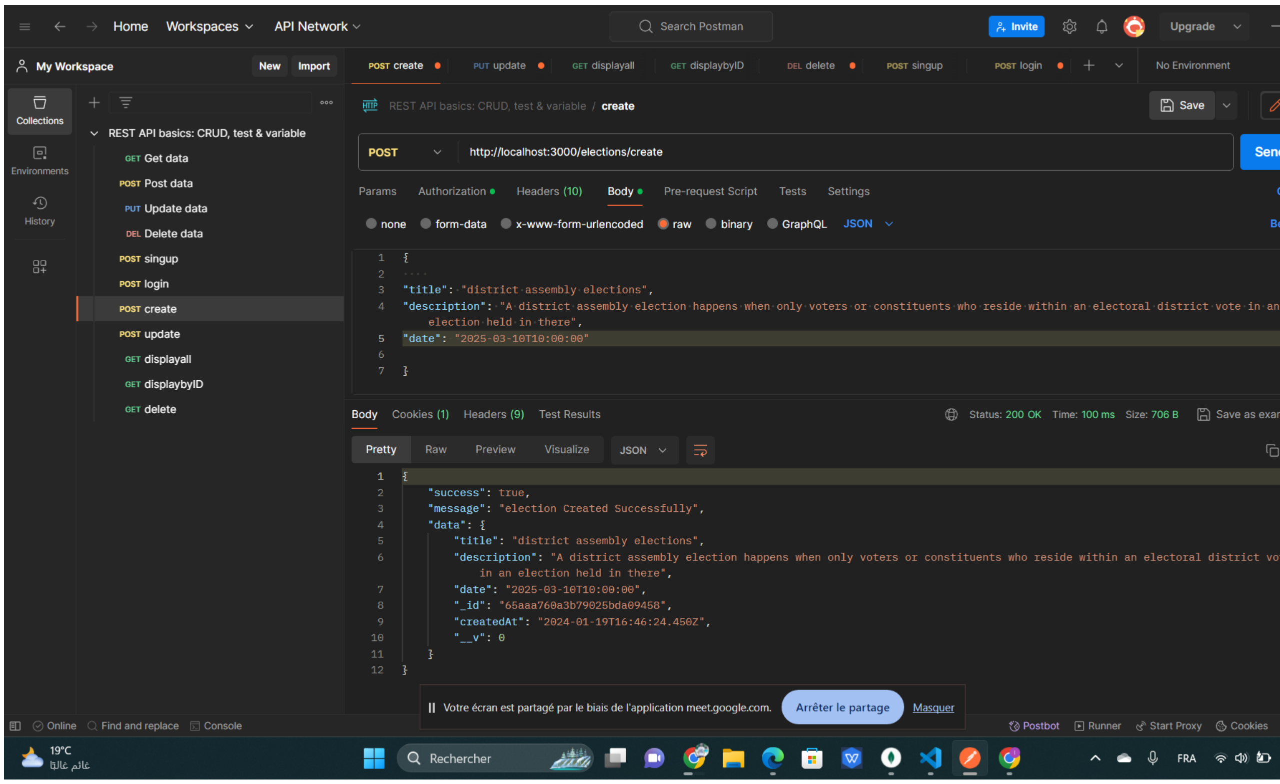
```
{
"email":"test1@gmail.com",
"password":"test123"
}
```

Status: 200 OK Time: 285 ms Size: 847 B

```
{
    "success": true,
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJfaWQiOiI2NWFhYThjM2EzYjc5MDI1YmRhMDk0NWQiLCJlbWFpbCI6InRlc3QxQGdtYWlsLmNvbSIsImlhdCI6MTcwNTY4MzM5NiwiZXhwIjoxNzA1NzA0OTk2fQ.
        5LC_tj5vcuLKr6Yxte7S8D3iHOSkS0uqyg2pAjjQUtE",
    "userId": "65aaa8c3a3b79025bda0945d"
}
```

4.3 Create Election

Endpoint: POST /elections/create
Description: Allows authenticated users to create a new election.
Actions: Validates input for title, description, and date.
Associates the election with the user ID.
Saves the election to the MongoDB database.



POST http://localhost:3000/elections/create

```
{

"title": "district assembly elections",
"description": "A district assembly election happens when only voters or constituents who reside within an electoral district vote in an
    election held in there",
"date": "2025-03-10T10:00:00"

}
```

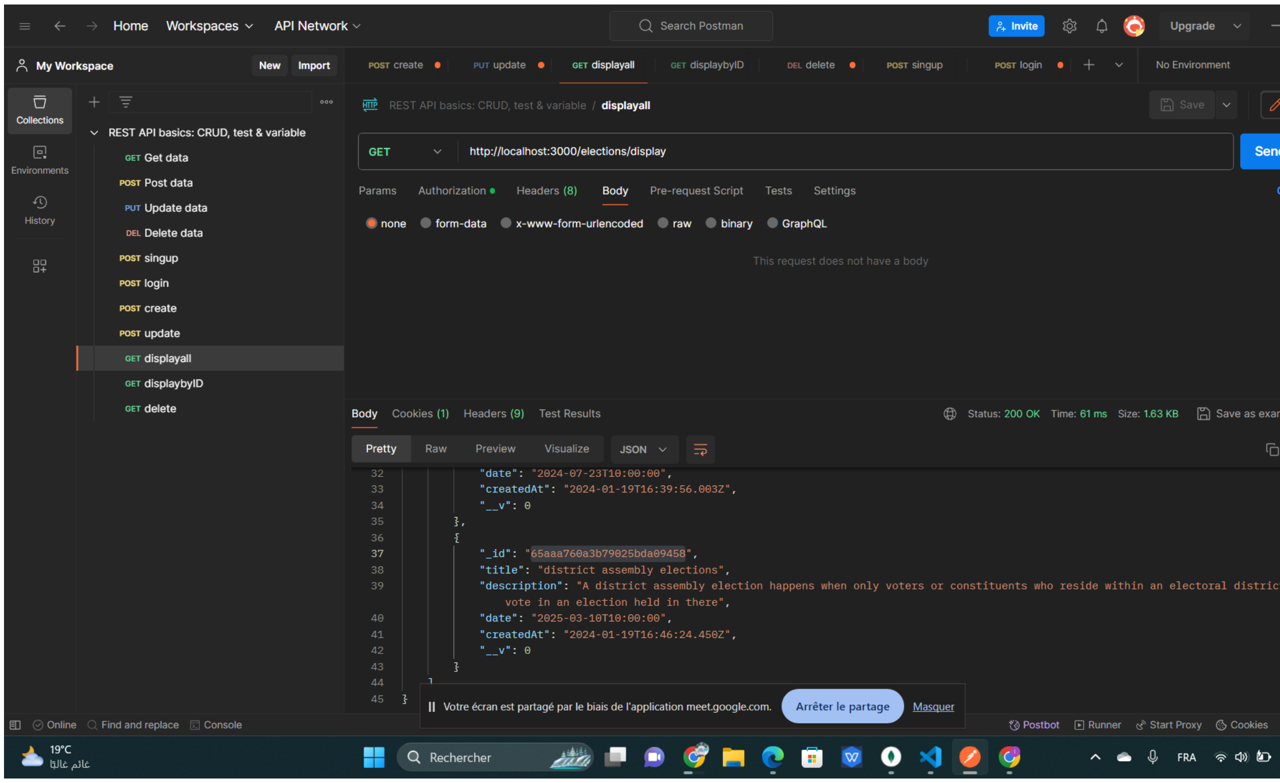
Status: 200 OK Time: 100 ms Size: 706 B

```
{
    "success": true,
    "message": "election Created Successfully",
    "data": {
        "title": "district assembly elections",
        "description": "A district assembly election happens when only voters or constituents who reside within an electoral district vo
            in an election held in there",
        "date": "2025-03-10T10:00:00",
        "_id": "65aaa760a3b79025bda09458",
        "createdAt": "2024-01-19T16:46:24.450Z",
        "__v": 0
    }
}
```

Votre écran est partagé par le biais de l'application meet.google.com. Arrêter le partage Masquer

4.4 Get All Elections

Endpoint: GET /elections/display
Description: Retrieves a list of all elections.
Actions: Queries the database for all elections.
Returns the list in the response.



GET http://localhost:3000/elections/display

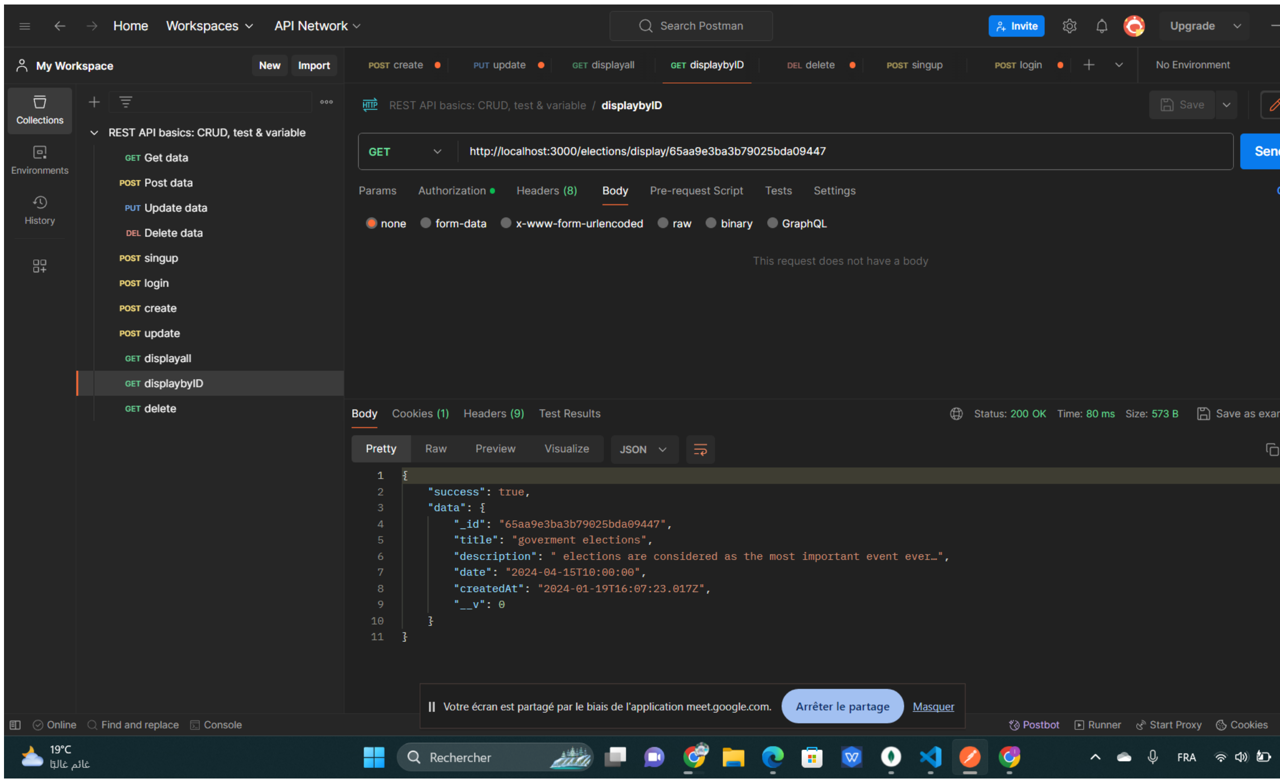
This request does not have a body

Status: 200 OK Time: 61 ms Size: 1.63 KB

```
            "date": "2024-07-23T10:00:00",
            "createdAt": "2024-01-19T16:39:56.003Z",
            "__v": 0
        },
        {
            "_id": "65aaa760a3b79025bda09458",
            "title": "district assembly elections",
            "description": "A district assembly election happens when only voters or constituents who reside within an electoral distric
                vote in an election held in there",
            "date": "2025-03-10T10:00:00",
            "createdAt": "2024-01-19T16:46:24.450Z",
            "__v": 0
        }
    ]
}
```

4.5 Get Election by ID

Endpoint: GET /elections/display/:electionId
Description: Retrieves a specific election by its ID.
Actions: Queries the database for the election with the specified ID.
Returns the election details in the response.



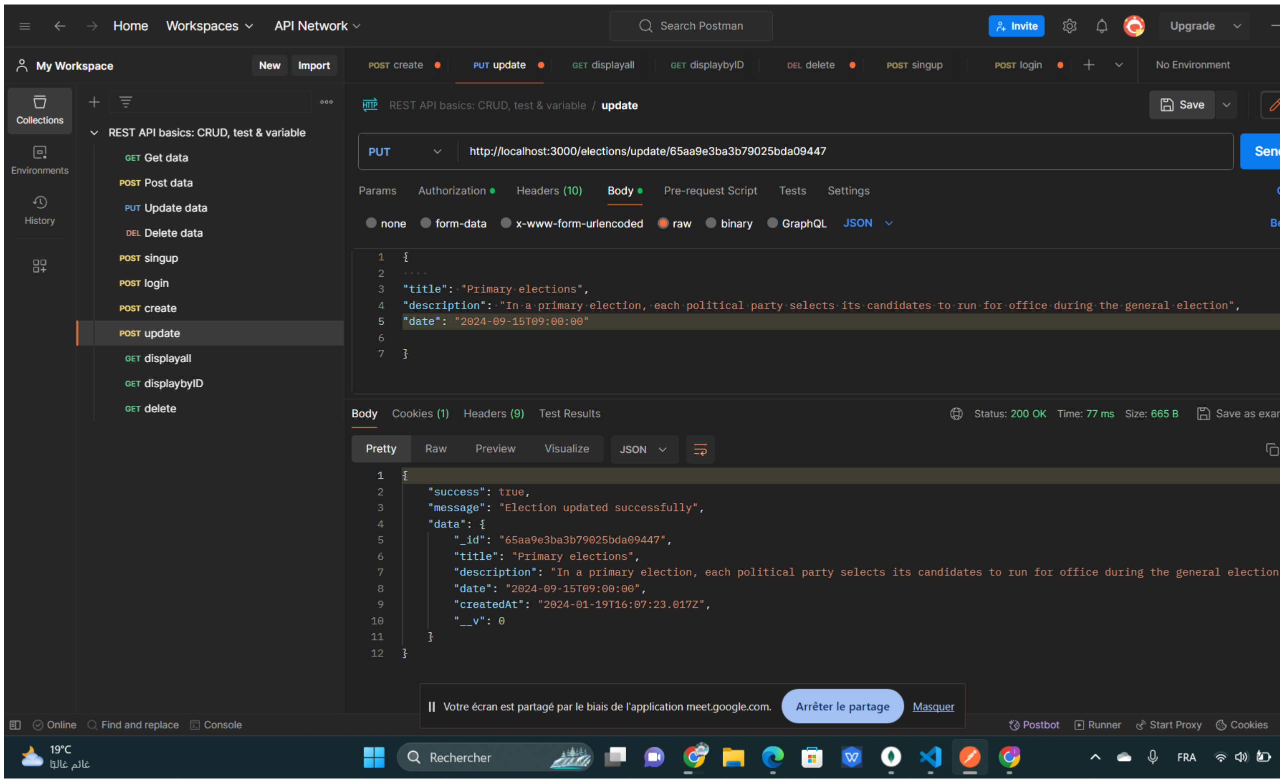
GET http://localhost:3000/elections/display/65aa9e3ba3b79025bda09447

Status: 200 OK Time: 80 ms Size: 573 B

```
{
    "success": true,
    "data": {
        "_id": "65aa9e3ba3b79025bda09447",
        "title": "goverment elections",
        "description": " elections are considered as the most important event ever…",
        "date": "2024-04-15T10:00:00",
        "createdAt": "2024-01-19T16:07:23.017Z",
        "__v": 0
    }
}
```

4.6 Update Election by ID

Endpoint: PUT /elections/update/:electionId
Description: Allows authenticated users to update an existing election.
Actions: Validates input for title, description, and date.
Updates the election in the database.
Returns the updated election details in the response.



PUT http://localhost:3000/elections/update/65aa9e3ba3b79025bda09447

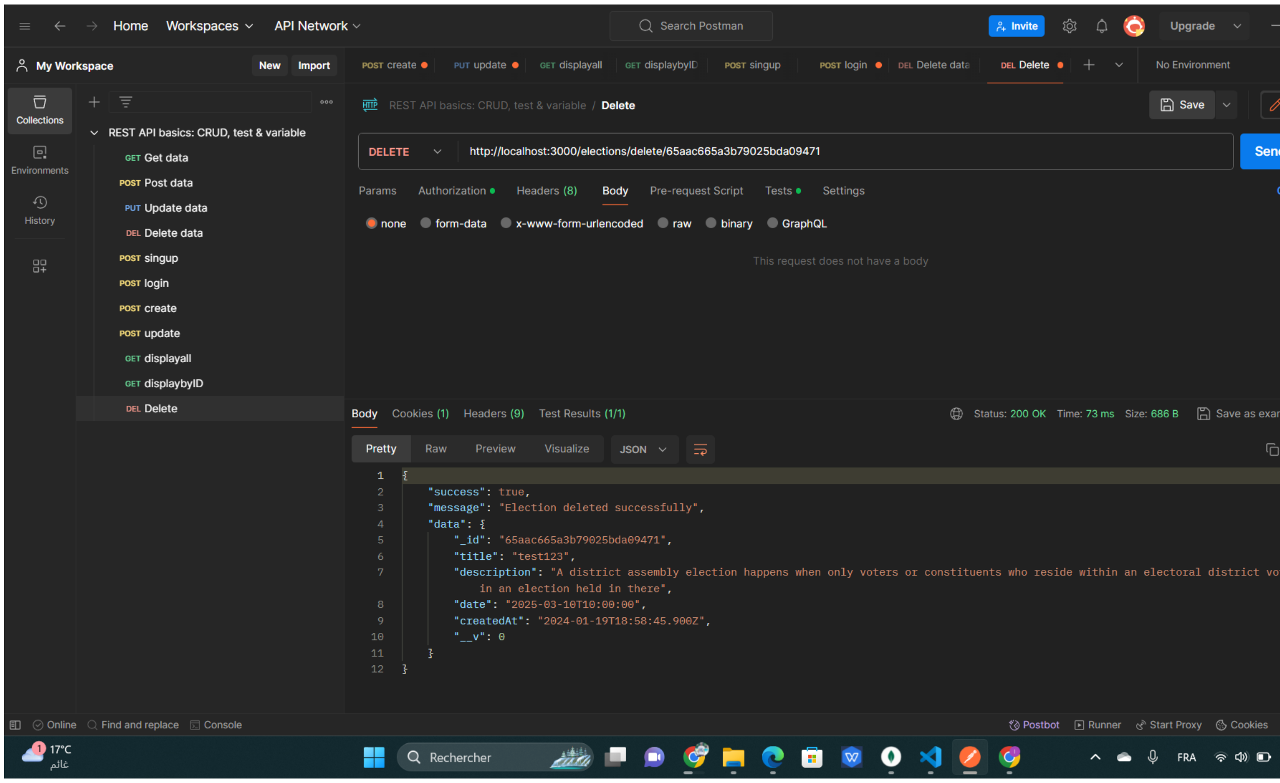
```
{
····
"title": "Primary elections",
"description": "In a primary election, each political party selects its candidates to run for office during the general election",
"date": "2024-09-15T09:00:00"

}
```

Status: 200 OK Time: 77 ms Size: 665 B

```
{
    "success": true,
    "message": "Election updated successfully",
    "data": {
        "_id": "65aa9e3ba3b79025bda09447",
        "title": "Primary elections",
        "description": "In a primary election, each political party selects its candidates to run for office during the general election
        "date": "2024-09-15T09:00:00",
        "createdAt": "2024-01-19T16:07:23.017Z",
        "__v": 0
    }
}
```

4.7 Delete Election by ID

Endpoint: DELETE /elections/delete/:electionId
Description: Allows authenticated users to delete an existing election.
Actions: Finds and deletes the election with the specified ID from the database.
Returns the deleted election details in the response.



DELETE http://localhost:3000/elections/delete/65aac665a3b79025bda09471

This request does not have a body

Status: 200 OK Time: 73 ms Size: 686 B

```
{
    "success": true,
    "message": "Election deleted successfully",
    "data": {
        "_id": "65aac665a3b79025bda09471",
        "title": "test123",
        "description": "A district assembly election happens when only voters or constituents who reside within an electoral district vo
            in an election held in there",
        "date": "2025-03-10T10:00:00",
        "createdAt": "2024-01-19T18:58:45.900Z",
        "__v": 0
    }
}
```

5 Conclusion

5.1 Achievements

The development and implementation of the Election Management System have achieved several key milestones. These achievements include:

- Successful integration of Node.js and Express.js for building a robust and scalable backend.
- Effective utilization of MongoDB as a NoSQL database, providing flexibility and high performance in managing diverse data types.
- Seamless incorporation of JWT for secure user authentication, enhancing the system's overall security.
- Implementation of user signup, login, and election management workflows, ensuring a user-friendly and functional system.
- Utilization of Postman for thorough API testing, contributing to the reliability of the endpoints.

5.2 Future Improvements

Improvements While the Election Management System has achieved notable success, there are areas where future improvements can be considered:

- Implementation of additional security measures, such as input validation and further encryption, to enhance overall system security.
- Exploration of additional features, such as user roles and permissions, to provide a more comprehensive and customizable user experience.
- Continuous monitoring and optimization of database queries to ensure optimal system performance, especially as the user base grows.
- Integration of logging and error handling mechanisms to facilitate efficient debugging and issue resolution.