**Today’s Agenda**

1. **Designing The VoterService Class**
2. **Designing The VoterController Class**
3. **Setting Up Postman And Testing Register Voter**
4. **Improving The Response**
5. **Testing Voter API**
6. **Modifying updateVoter()**
7. **Designing The VoterService Class**

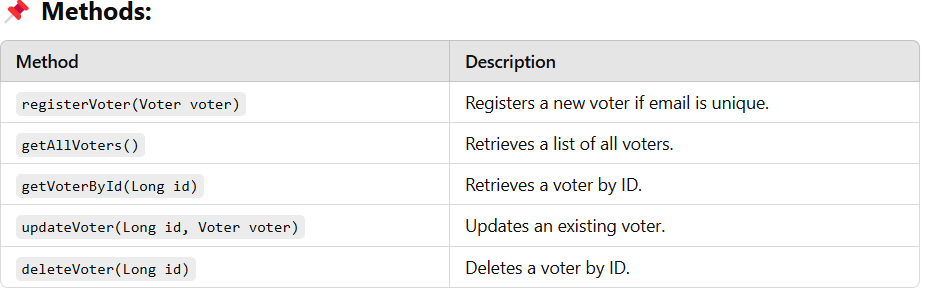
**1. VoterService.java**

**📌 Explanation**

**Purpose:** **Implements** the **business logic** for **voter-related operations**.

**📌 Annotations Used:**

* **@Service** → **Marks** this **class** as a **Spring service** component.



**Code For VoterService Class**



1. **Designing The VoterController Class**

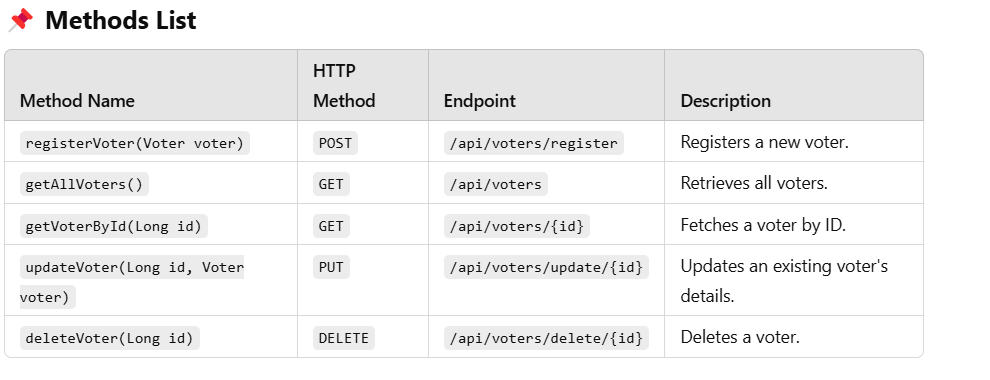
**1. VoterController.java**

**📌 Explanation**

**Purpose**: Exposes **REST API endpoints** for managing **voter-related operations**.

**📌 Annotations Used:**

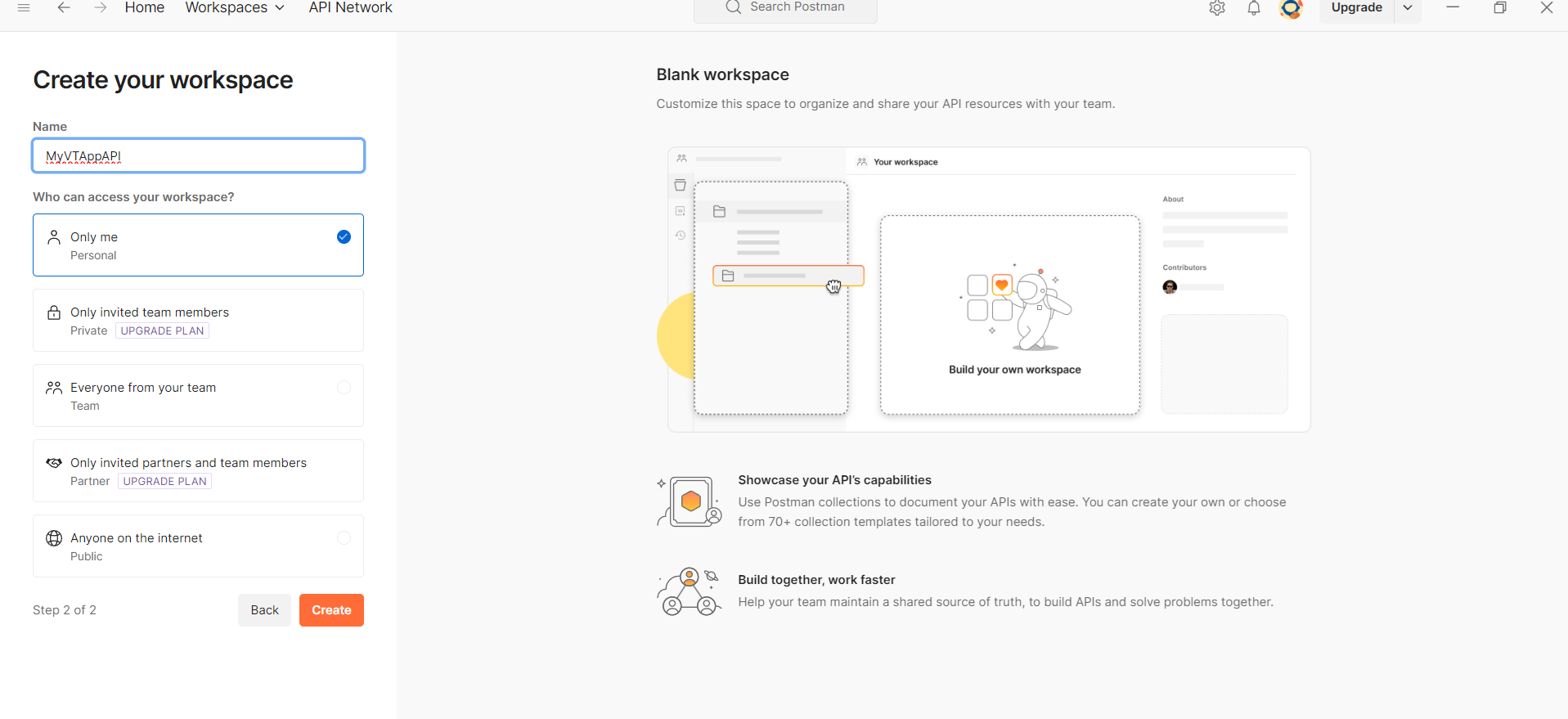
* **@RestController** → **Marks** this **class** as a **REST API controller**.
* **@RequestMapping("/api/voters")** → **Base path** for all **voter-related** APIs.
* **@PostMapping, @GetMapping, @PutMapping, @DeleteMapping** → **Defines HTTP endpoints** for **CRUD operations**.

****

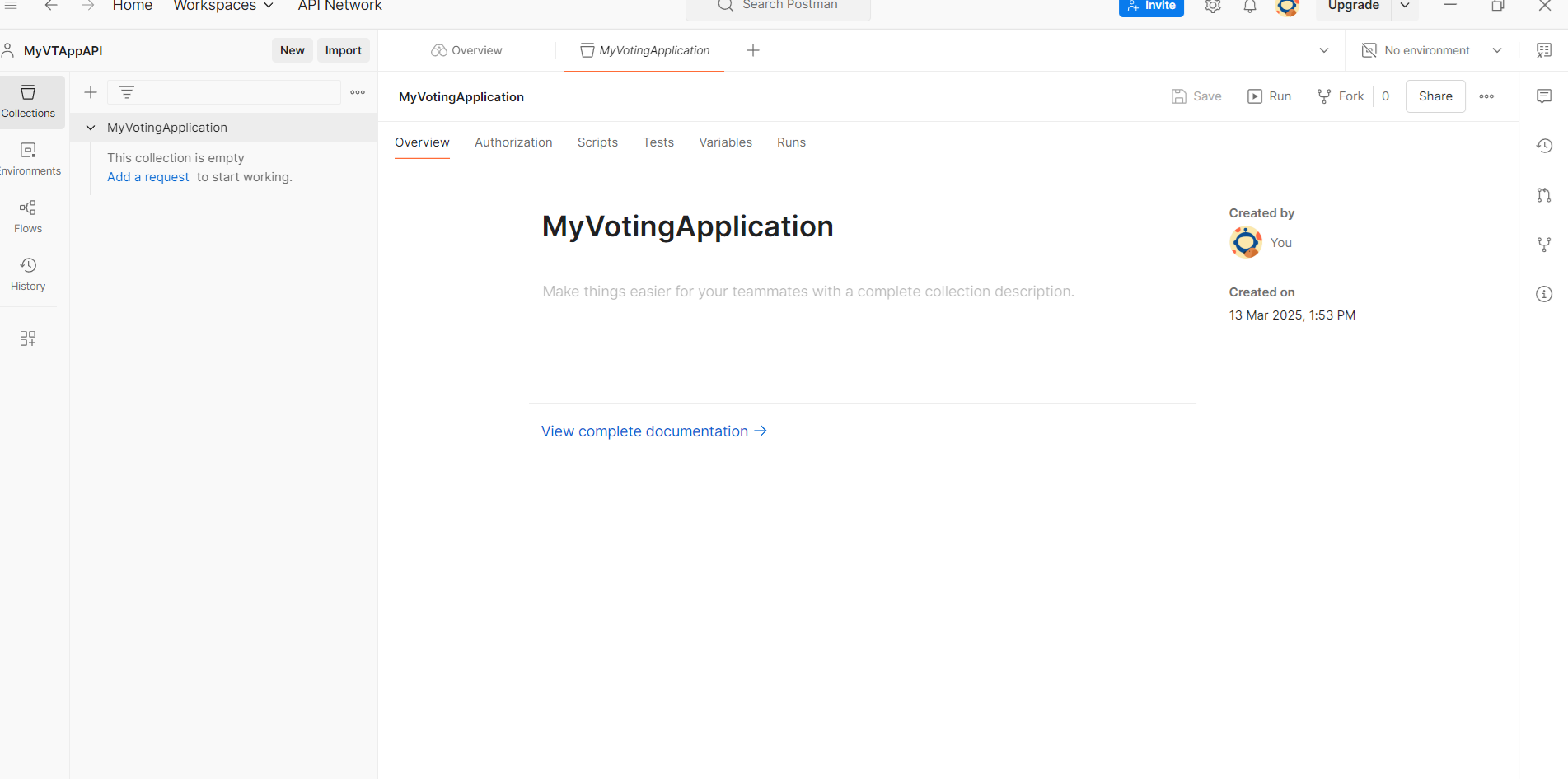
**Code For VoterController Class**

****

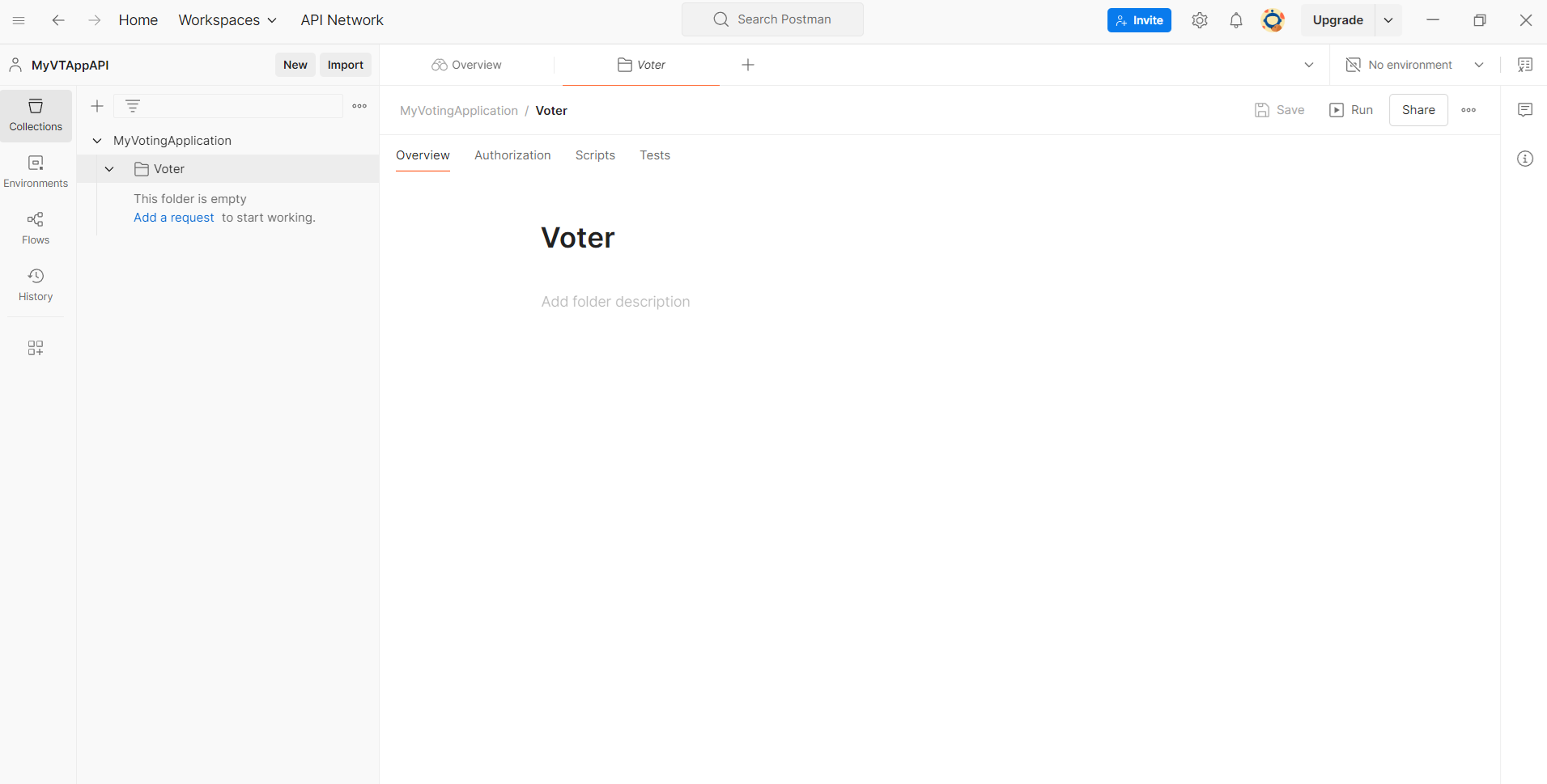
1. **Setting Up Postman And Testing Register Voter**
2. **Login to postman on your** desktop app **and click on** workspaces **and then on Create WorkSpace**
3. **Click** on **Next**
4. **Fill** the **workspace name** and **click** on **create**

****

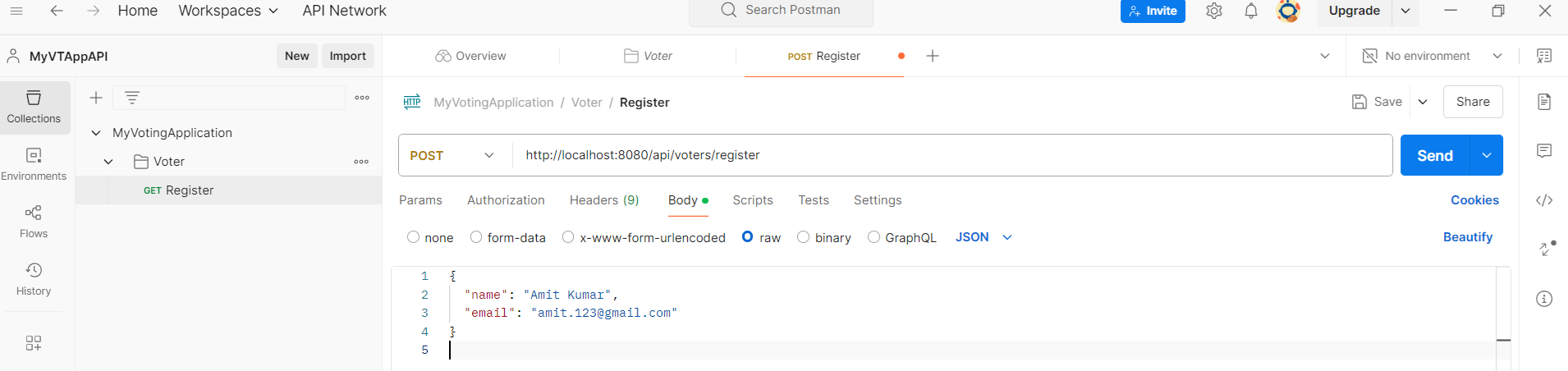
1. Now we will **create collection**. So click on **Collection** and then **click** on **create collection**
2. Give **some name** to the **Collection**.

****

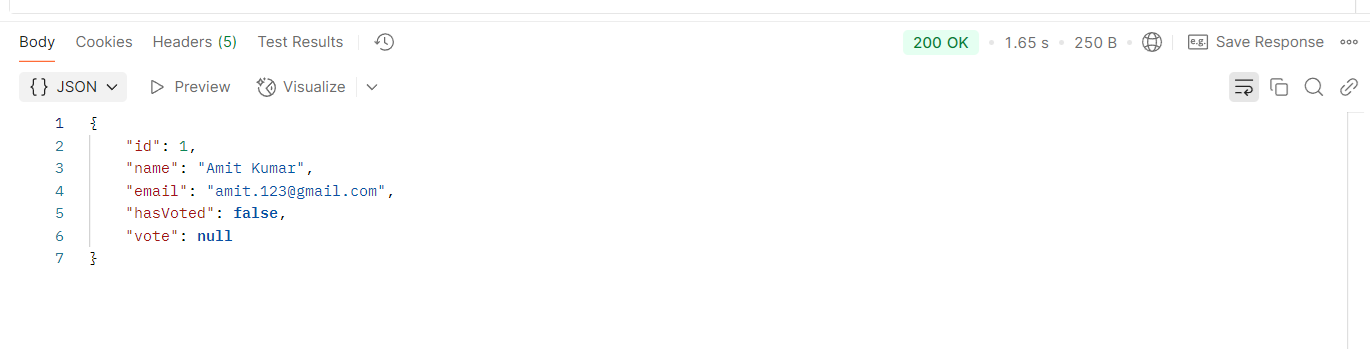
1. **Click** on . . . And select **Add Folder** and then **rename** it.



1. **Click** on . . . And select **Add Request** .
2. Now set it to **POST** and rename it to **register**.
3. Now **test it** by **sending** the **request**

****

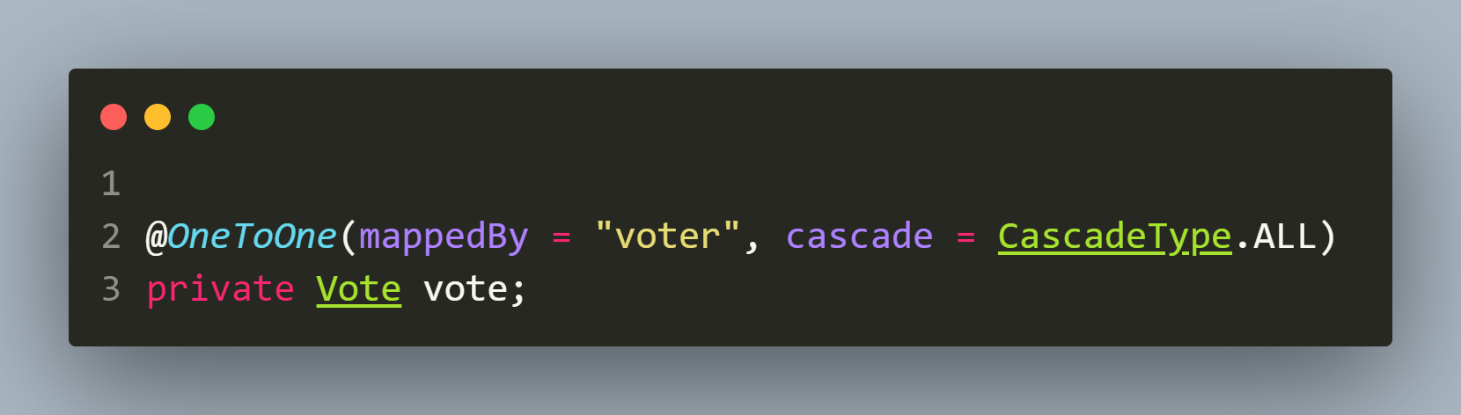
1. You **should receive** the **following response**

****

1. **Improving The Response**

**1.** You must have **noticed** that we got **“vote”:null** in the **response**.

**2. That’s because** the **Voter** entity has a **one-to-one relationship** with **Vote**:

****

3. When a **new voter** is **registered**, they **haven’t cast a vote** **yet**, so the **vote field** remains **null**.

4. The **JSON response** automatically includes the **vote** field, but ***since no vote is linked to the voter***, it returns **null**.

📌 **How to Fix or Handle This?**

* + - 1. Ignore **vote** in **JSON Response** (If Not Needed Initially)
      2. This **can be done** by using **@JsonIgnore** on the **vote** field in the **Voter** entity.

📌 **What is @JsonIgnore ?**

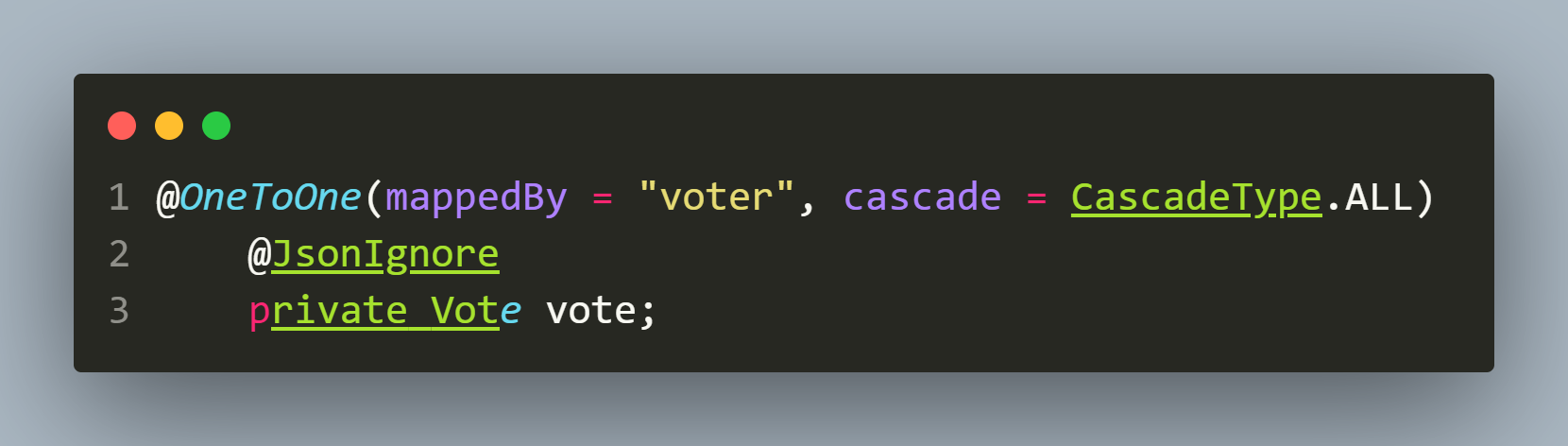
**@JsonIgnore** is a **Jackson annotation** (from the **com.fasterxml.jackson.annotation** package) that tells **Spring Boot**:

**"When converting this object to JSON, ignore this field."**

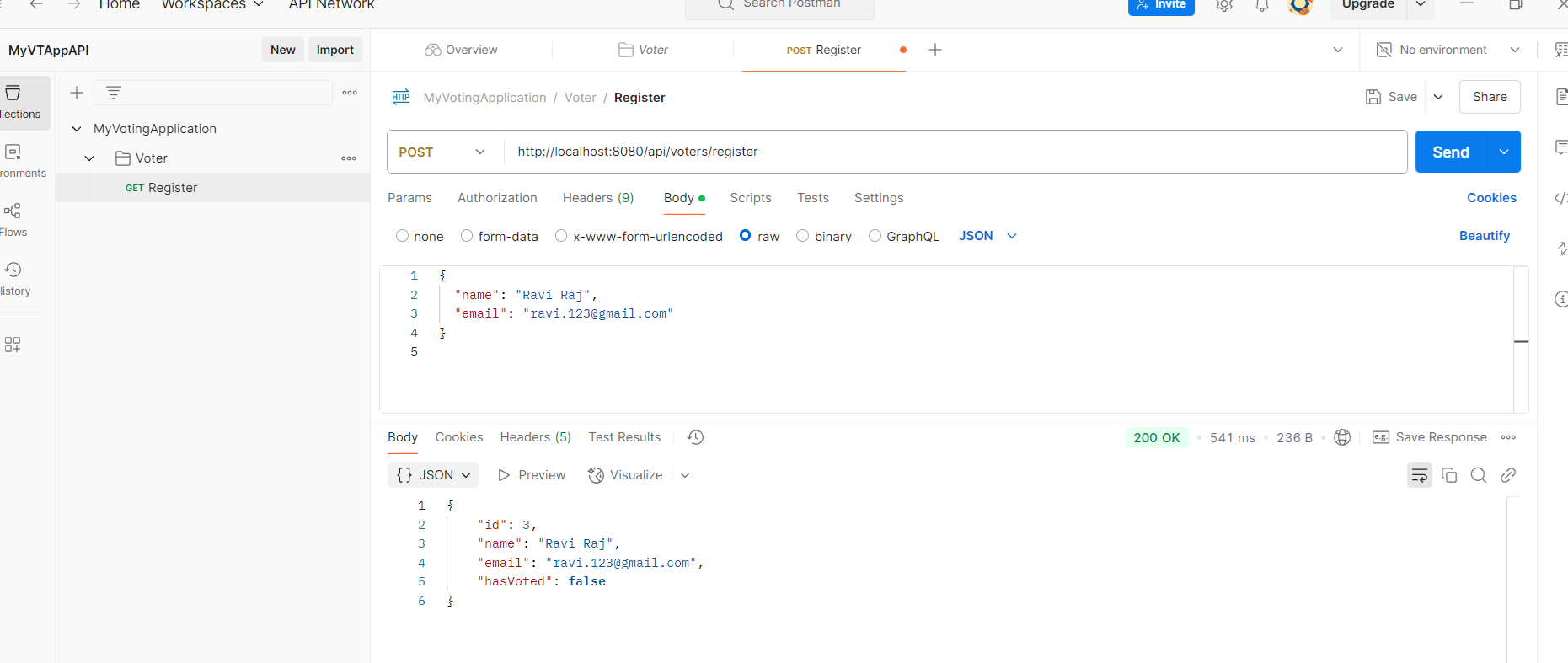
**In short:**

* **It hides a field** from **being included** in **JSON output** (when **sent** from **backend** to **frontend**).
* It’s **useful** when you want to **prevent certain data from being exposed** in your **REST APIs**.

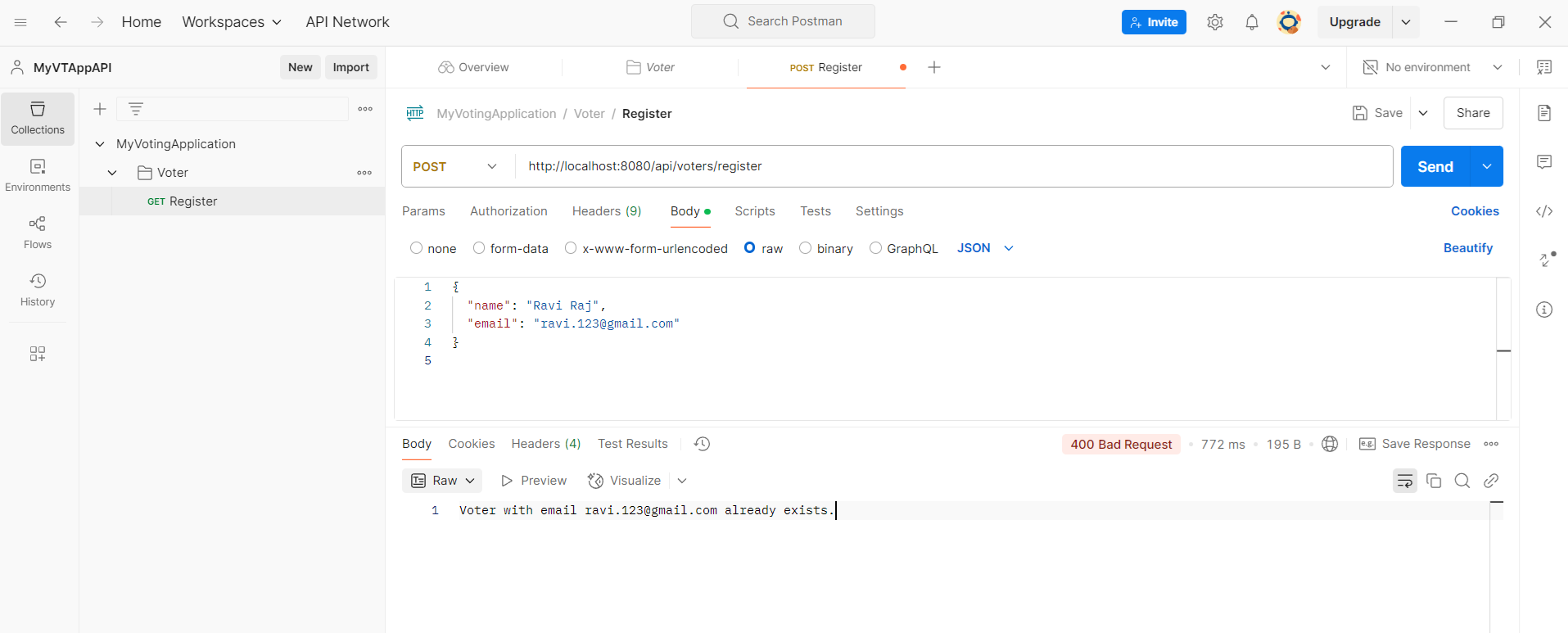
📌 **Modified Code**

****

3.Now **test it again**  by **sending** the **request**

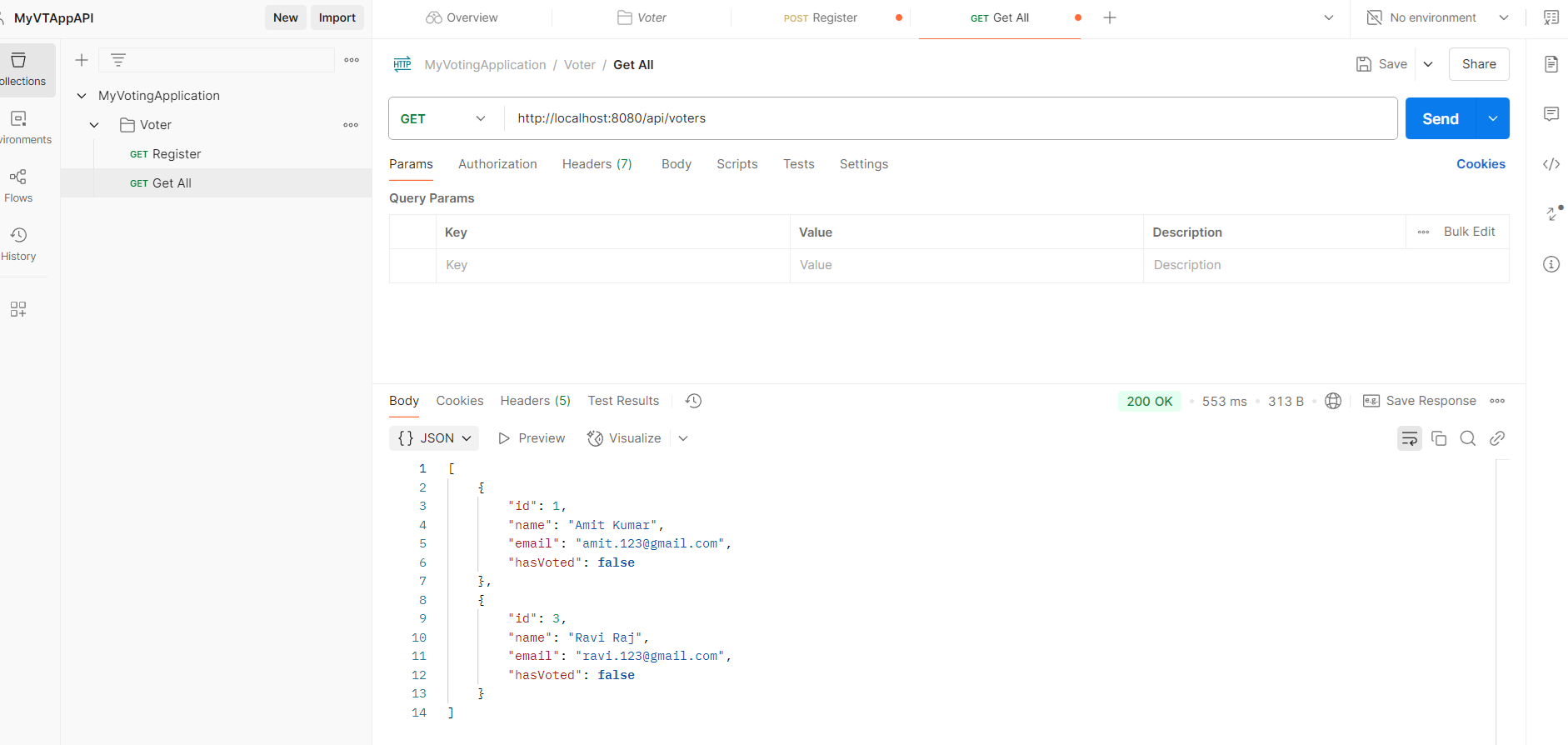
****

4.Try **giving** the **same request data again** and you will see the **error message**

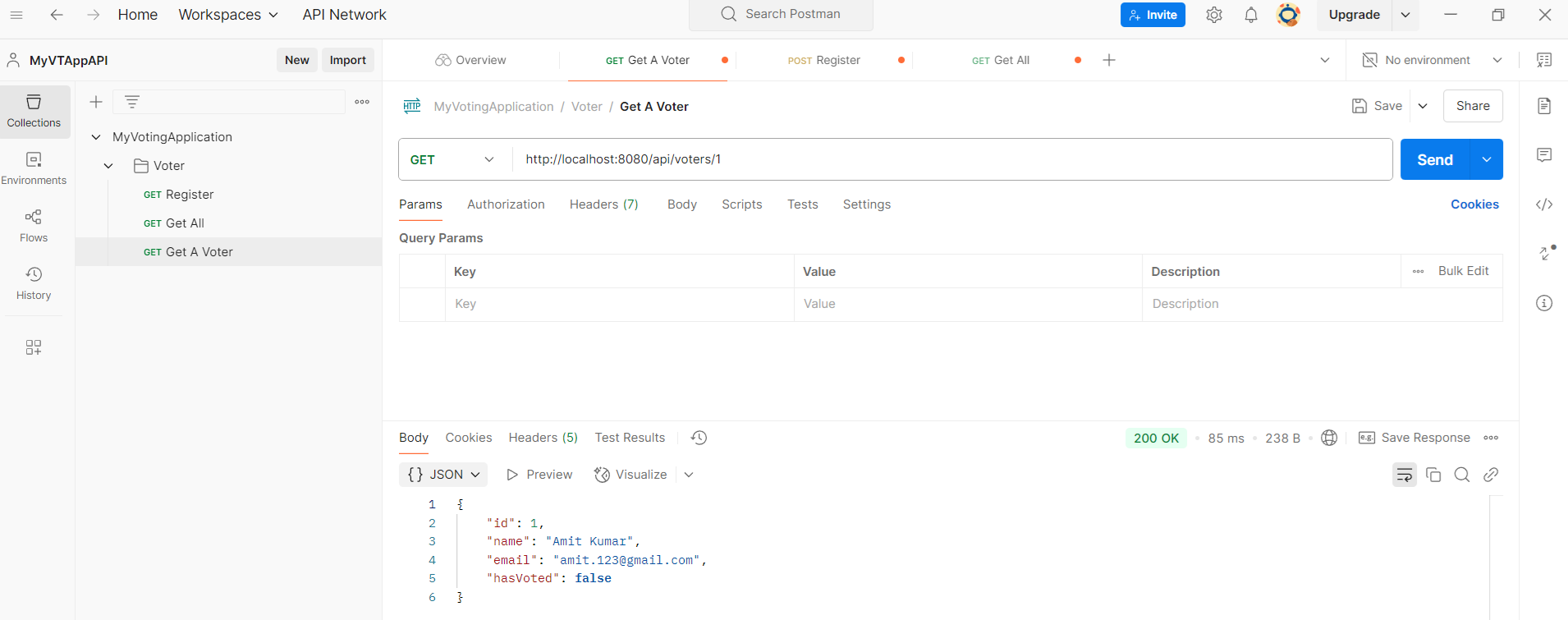
****

**5.Testing The Voter API**

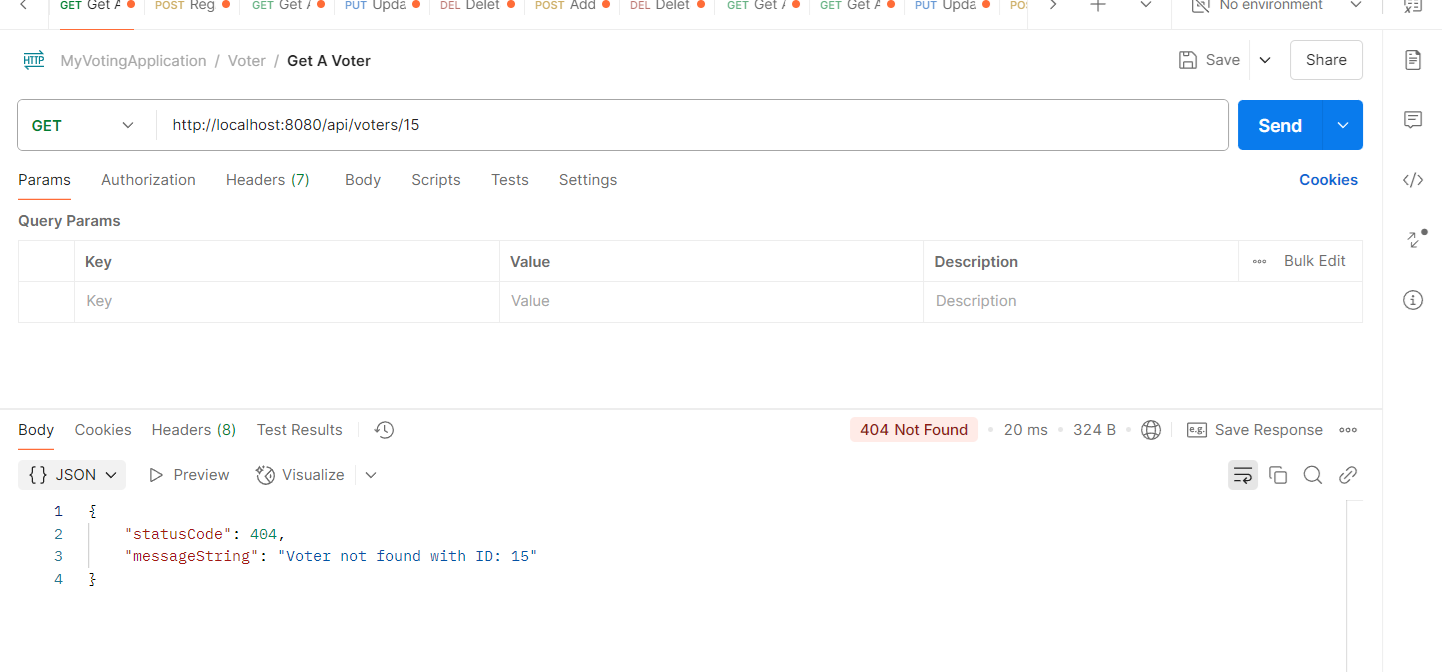
1. **Get All Voters: *http://localhost:8080/api/voters***

****

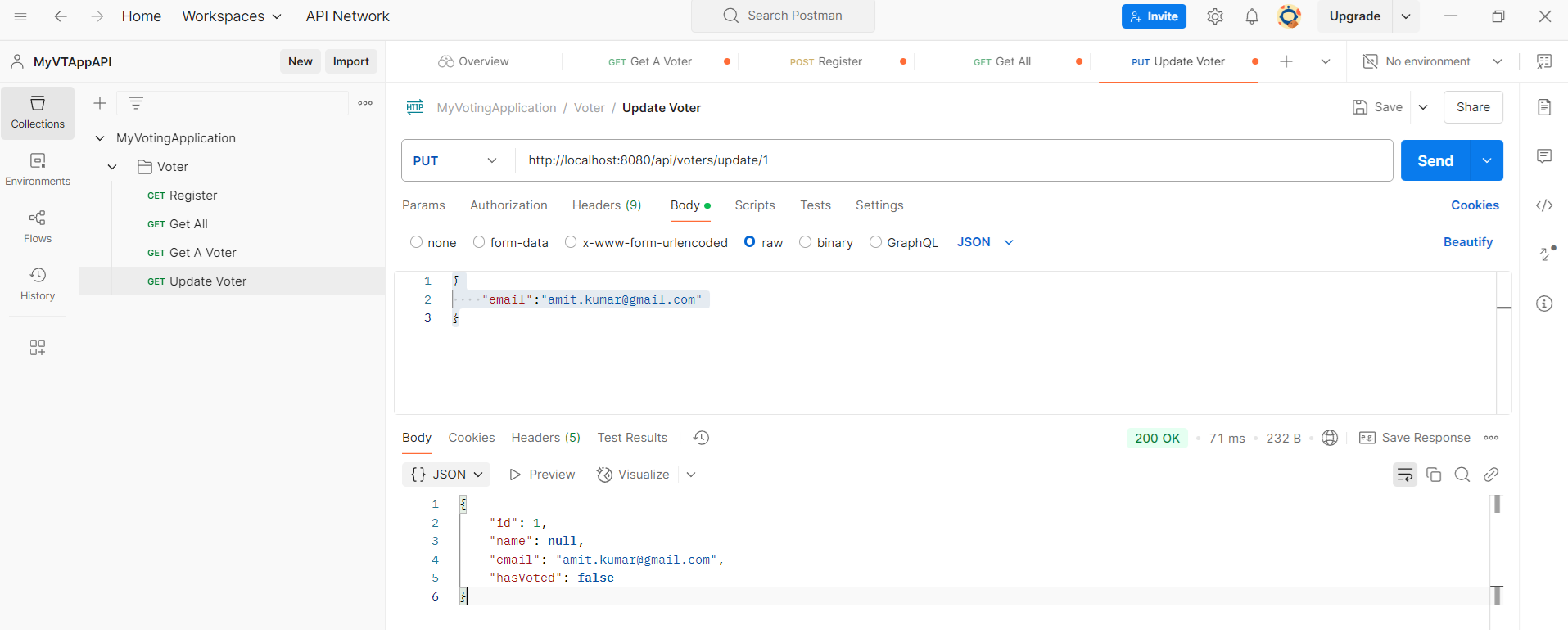
1. **Get Voter By Id:** [***http://localhost:8080/api/voters***](http://localhost:8080/api/voters)***/1***

******

1. **Getting A Non Existing Voter :** [***http://localhost:8080/api/voters/15***](http://localhost:8080/api/voters/15)

******

1. **Update A Voter :** [***http://localhost:8080/api/voters/1***](http://localhost:8080/api/voters/1)

******

**Can you figure out a problem in the above response ?**

1. Although the **object** being **updated** has **name** filed set **but still** we are getting **null**.
2. **That’s because** the method **replaces the existing voter with the new data** from the **request** **body**.
3. Our **request body** only contains:

**{**

**"email": "amit.kumar@gmail.com"**

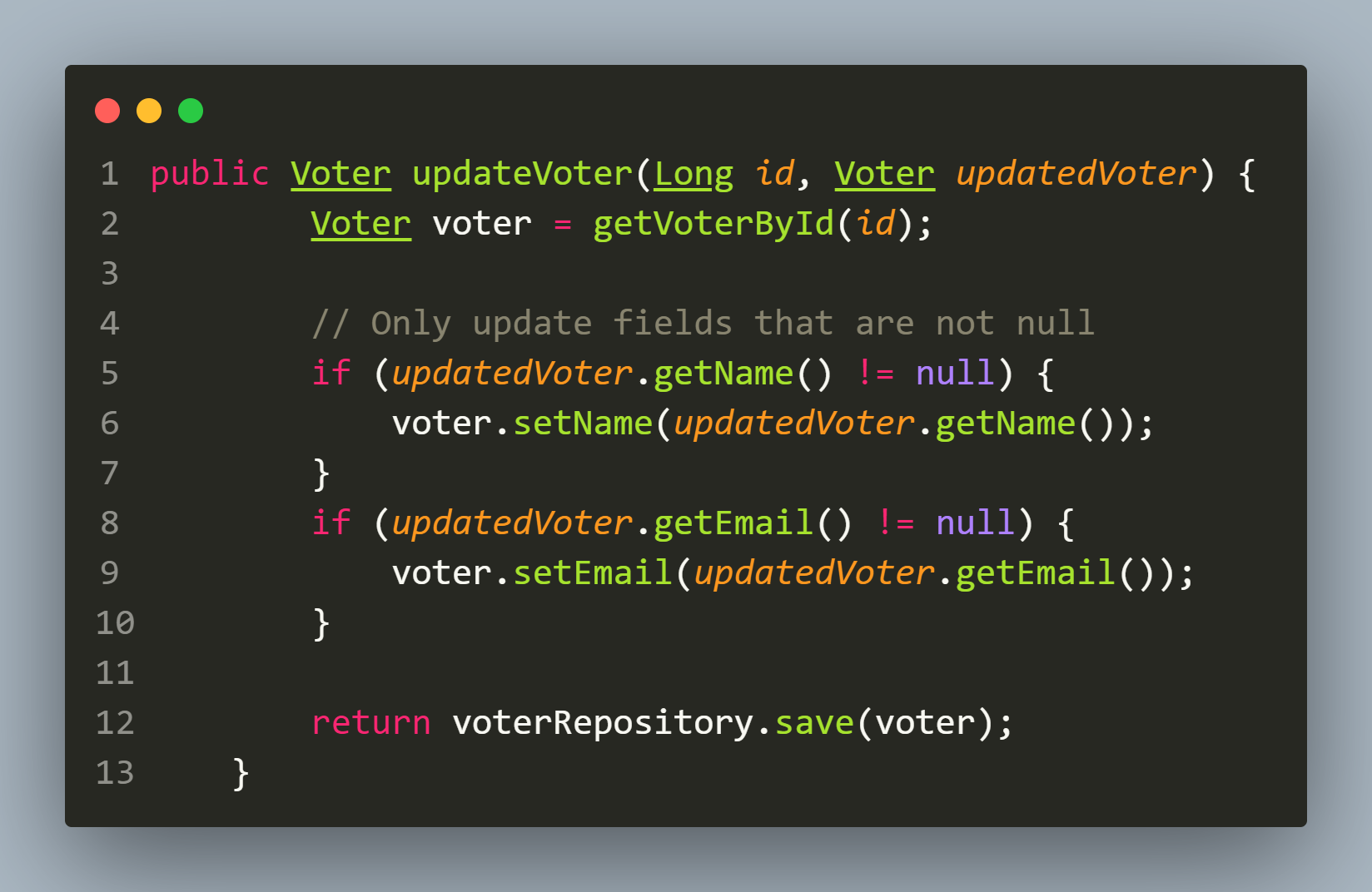
**}**

1. Since **name** is **not included**, **Spring** sets it to **null** when **updating** the entity.

**Can you find the solution ?**

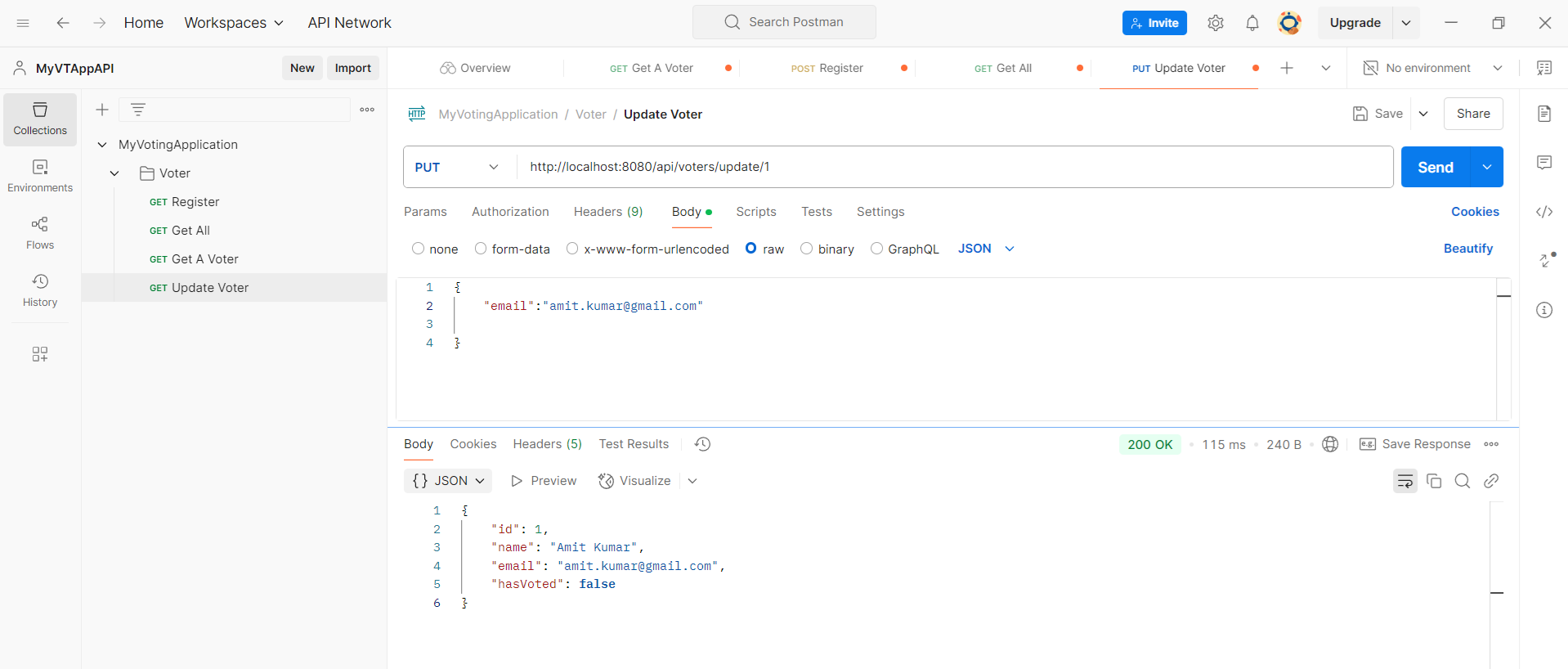
If we **want to update only the provided fields** (*without setting missing fields to null*), modify the **updateVoter()** method of **VoterService** class .

**6. Modifying The updateVoter()Testing The Voter API**

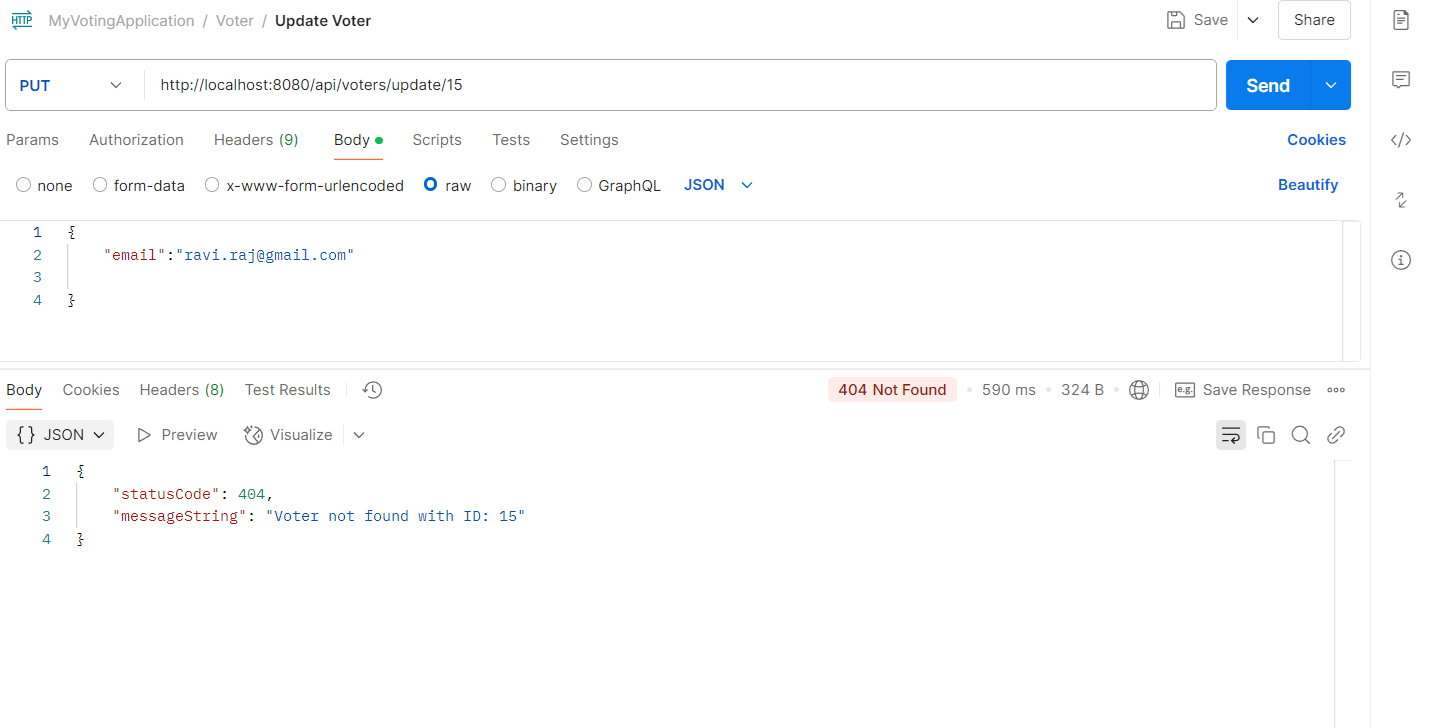
****

**Test again**

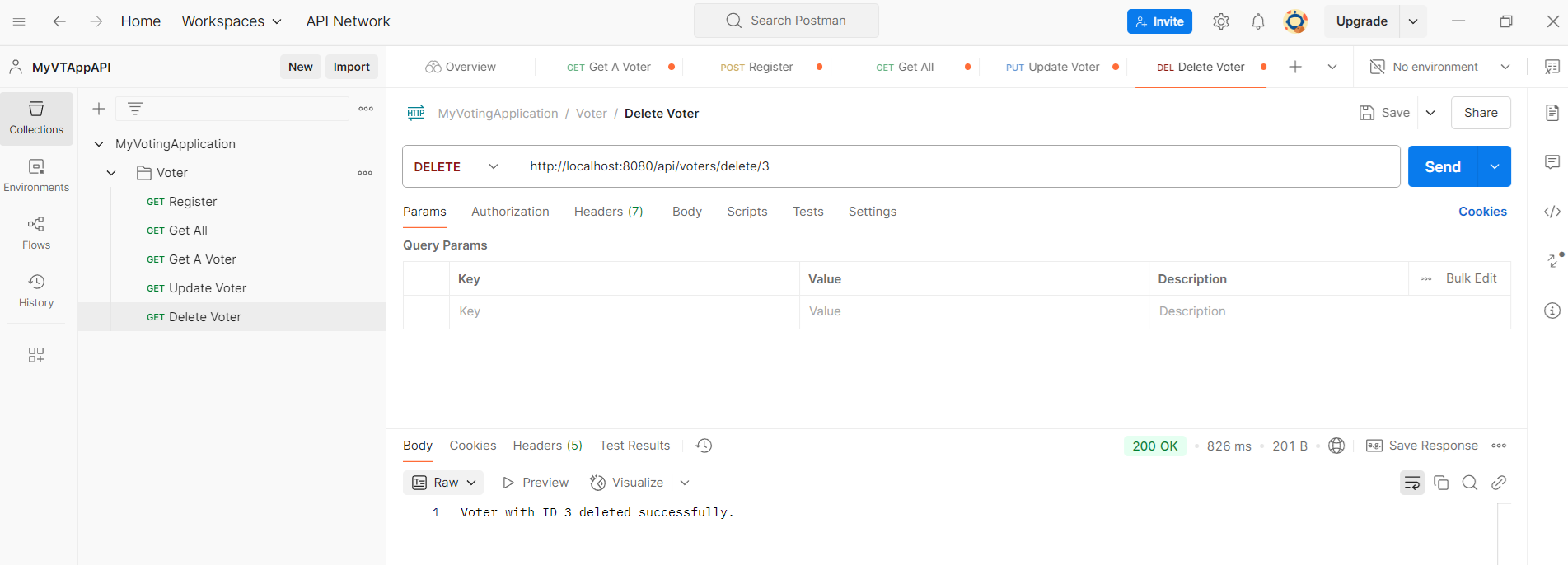
**3. Update A Voter :** [***http://localhost:8080/api/voters/1***](http://localhost:8080/api/voters/1)

****

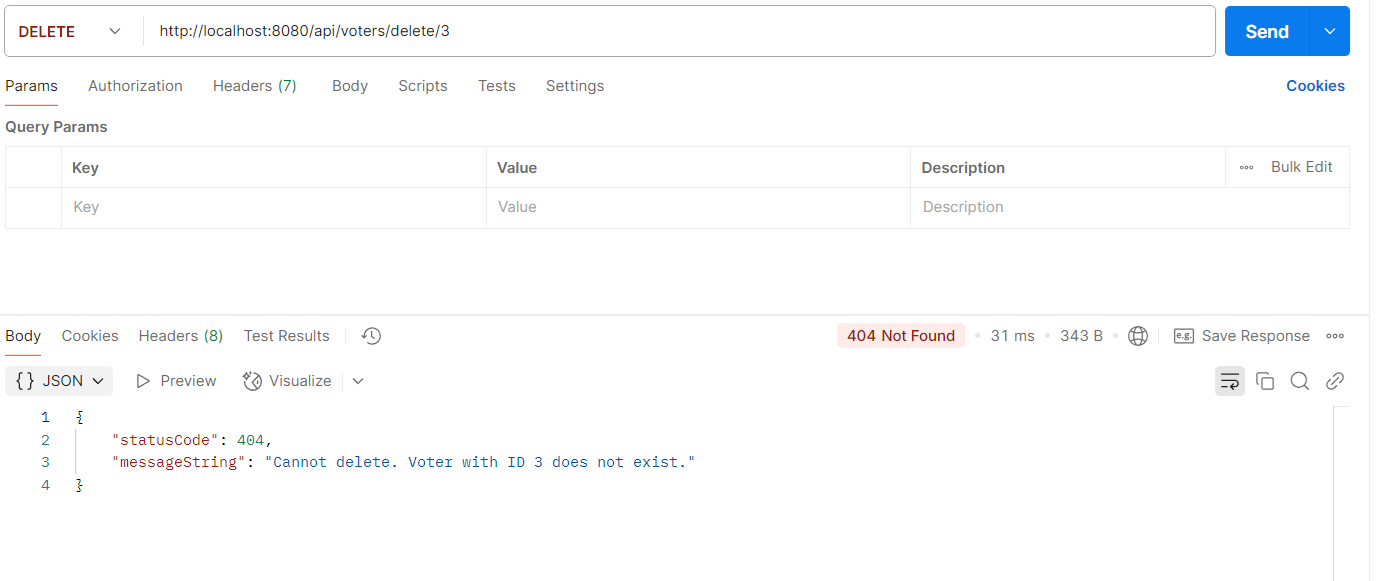
**Updating non-existing voter:** [***http://localhost:8080/api/voters/update/4***](http://localhost:8080/api/voters/update/4)

**

**4. Delete A Voter :** [***http://localhost:8080/api/voters/delete/3***](http://localhost:8080/api/voters/delete/3)

**

**Deleting non-existing voter:** [***http://localhost:8080/api/voters/delete/3***](http://localhost:8080/api/voters/delete/3)

**