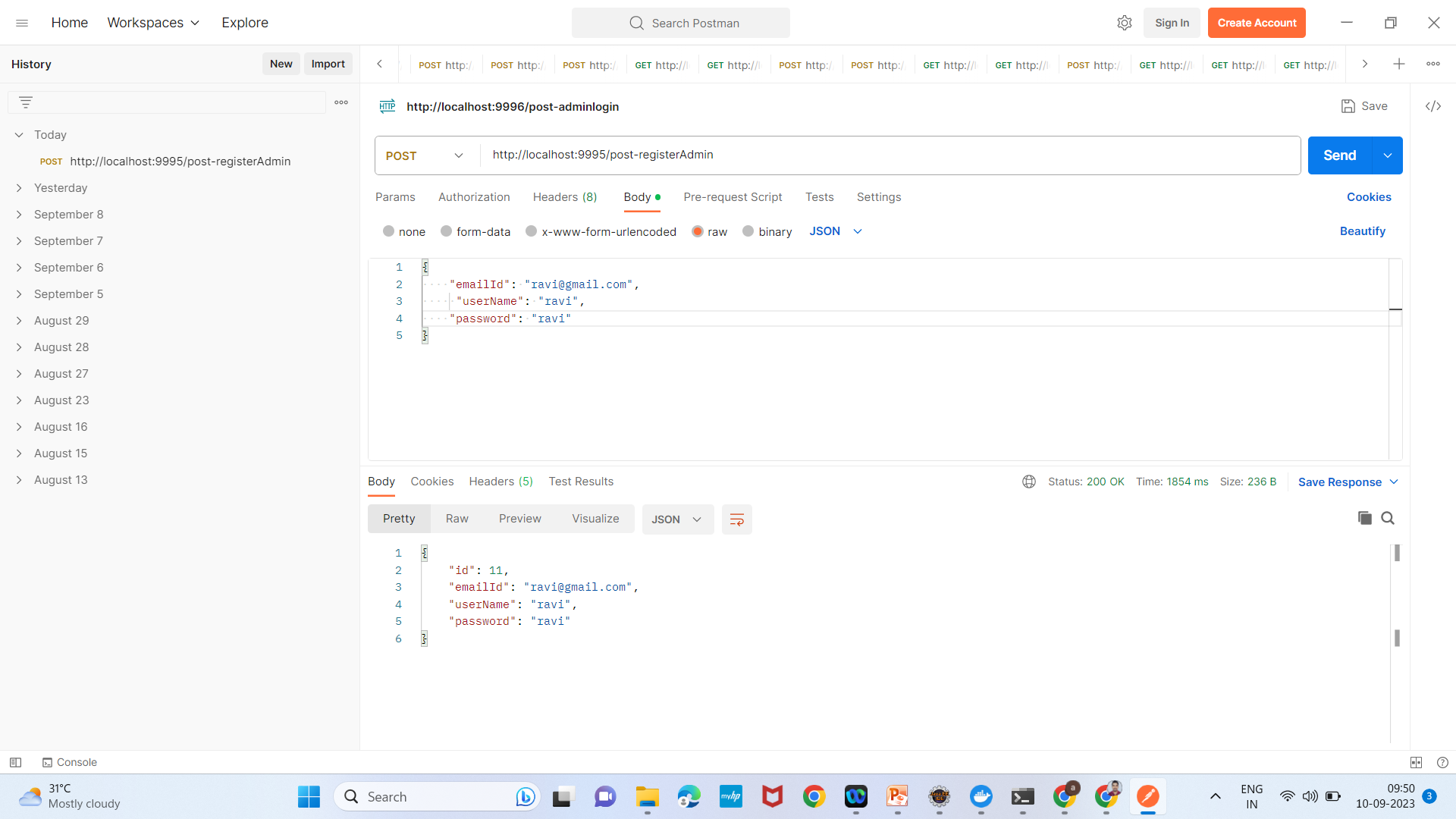
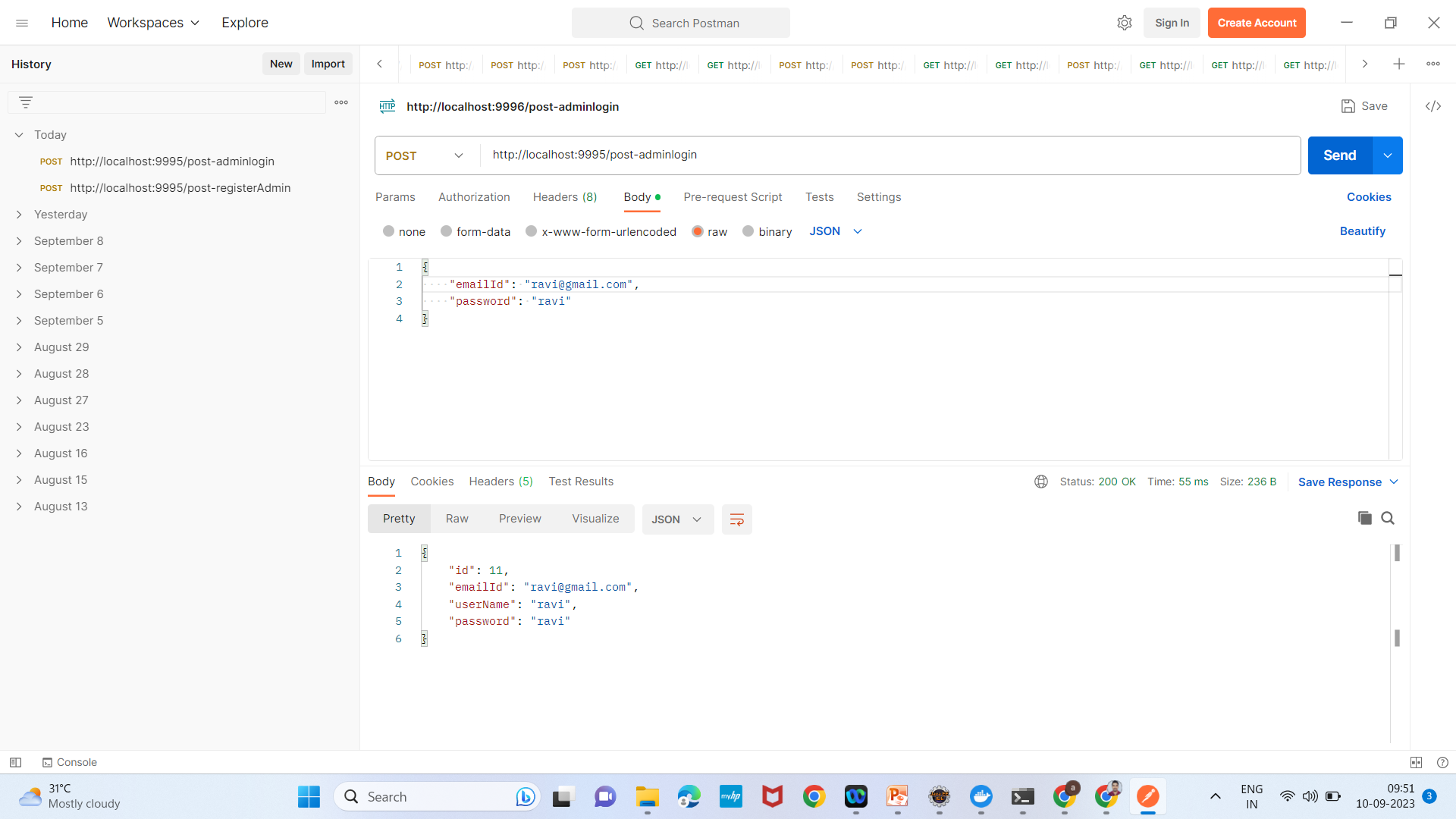
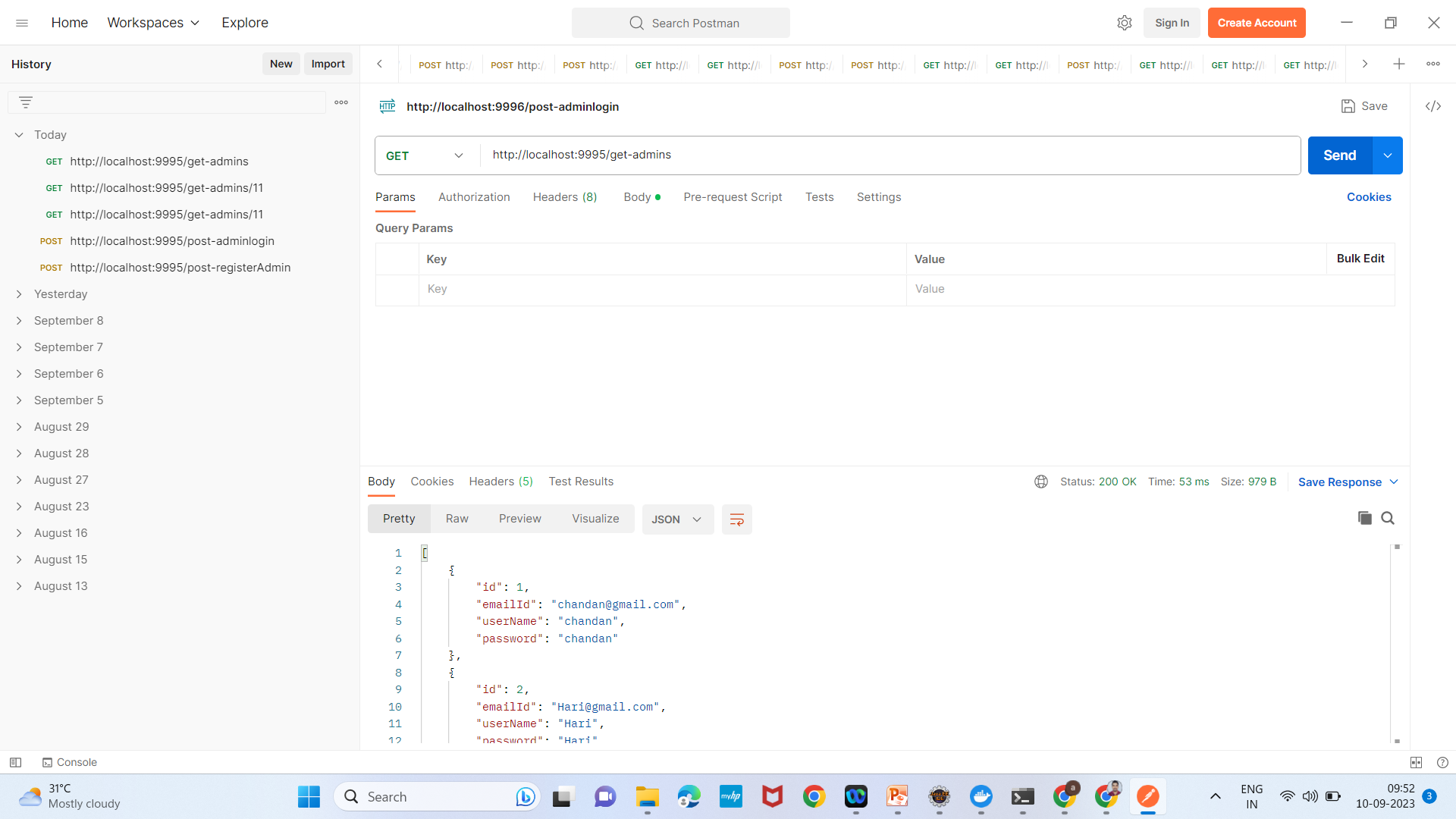
|  |  |
| --- | --- |
| SERVICES | EXPLAINATION |
| Admin | * **Save Admin**: This operation is handled by the saveAdmin method It is used to save admin details with admin\_id , username, * Email-id and Password . * **Get All Admins**: This operation is handled by the getAdmin method and It retrieves a list of all admin. * **Check Admin Login**: This operation is handled by the checkAdmin method and It takes an Admin object in the request body, checks if the provided admin credentials are valid or not. * **Get Admin by ID**: This operation is handled by the getAdminById method and It takes an admin ID , fetches the corresponding admin. |
| Doctor Signup and Login | * **Get All Doctors**: This operation is handled by the getAllDoctors method and it retrieves a list of all doctors. * **Get Doctor by ID**: This operation is handled by the getDoctorById/{d\_id} method and it takes a doctor’s ID and fetches the corresponding student. * **Add Doctor**: This operation is handled by the addDoctor method. It is used to save doctor details with the doctor’s name, gender, password, contact n0, specialization, experience, salary, and acceptance status details. * **Update Doctor**: This operation is handled by the updateDoctor method. It is used to update details of the doctor’s name, gender, password, contact n0, specialization, experience, salary, and acceptance status details by providing the doctor’s unique ID. * **Delete Doctor**: This operation is handled by the deleteDoctor method. It is used to delete the doctor from the list. * **PostMapping**("/doctorlogin"): Specifies that this method should handle POST requests at the "/doctorlogin" endpoint. |
| Discharged Patients | * **Get All Discharged Patients**: This operation retrieves a list of all discharged patients in the system. * **POST Method**: Add Discharged Patient. This operation is used to add a discharged patient to the system. |
| Patient Signup and Login | * **Save Patient Sign-Up**   Method: addPatientSignUp  Purpose: Saves patient registration details, including patient ID, name, username, password, gender, and contact number.   * **Get All Patients**   Method: getAllPatients  Purpose: Retrieves a comprehensive list of all registered patients.   * **Update Patient Sign-Up**   Method: updatePatientSignUp  Purpose: Permits updates to patient registration information, such as name, username, password, gender, and contact number.   * **Delete Patient Sign-Up**   Method: deletePatient  Purpose: Removes a patient's registration record based on the provided patient ID.   * **Check Patient Login**   Method: checkPatientLogin  Purpose: Verifies the authenticity of patient login credentials by comparing provided username and password against the stored records.   * **Get Patient by ID**   Method: getPatientById  Purpose: Retrieves specific patient registration details based on the patient's unique ID. |
| Appointments | * @**GetMapping("/appointments"):**   Inside the method, it calls appointmentrepository.findAll() to retrieve a list of all appointments from the database.   * @**GetMapping("/appointments/{ap\_id}"):**   Inside the method, it uses appointmentrepository.findById(ap\_id) to attempt to find an appointment by its ID.   * **PutMapping("/appointments/{ap\_id}"):**   Inside the method, it attempts to find the appointment by its ID using appointmentrepository.findById(ap\_id).   * **@PostMapping("/appointments"):** * It saves the new appointment to the database using appointmentrepository.save(a) and returns the saved appointment in the response. * **@DeleteMapping("/appointments/{ap\_id}"):**   Inside the method, it attempts to find the appointment by its ID using appointmentrepository.findById(ap\_id).  If the appointment with the specified ID exists, it is deleted from the database using appointmentrepository.delete(a). |
| Feedbacks | * **@GetMapping("/feedbacks"):**   Inside the method, the code calls feedbackrepository.findAll(), which is likely a method provided by the FeedbackRepository (a Spring Data JPA repository) to retrieve all feedback entries from a database.  **@PostMapping("/feedbacks"):**   * Inside the method, the code calls feedbackrepository.save(f), which is likely a method provided by the FeedbackRepository to save the received Feedback object to the database. |
| **SERVICES** | **SCREENSHOTS** |

**Admin Screenshots:**

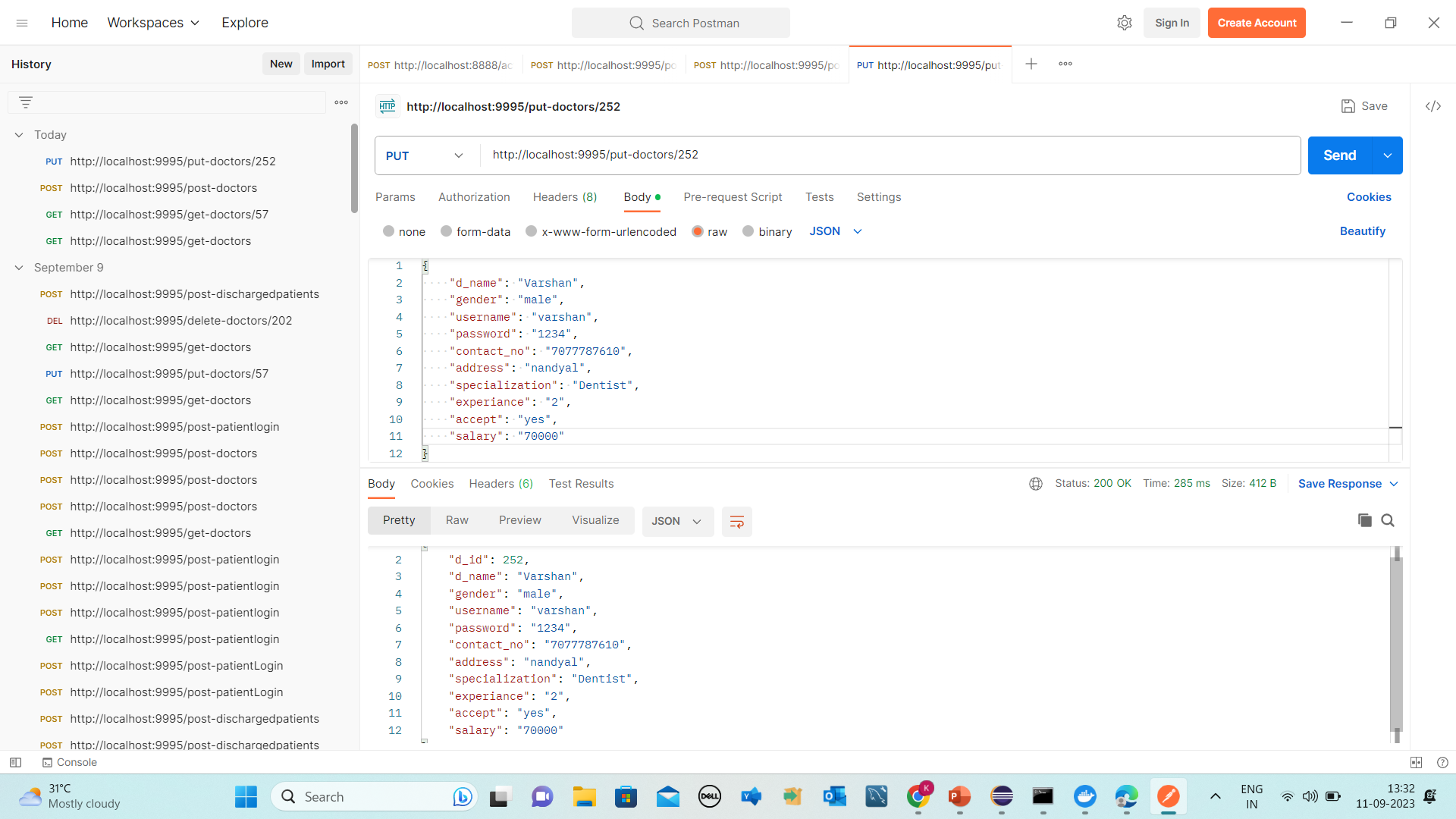
****

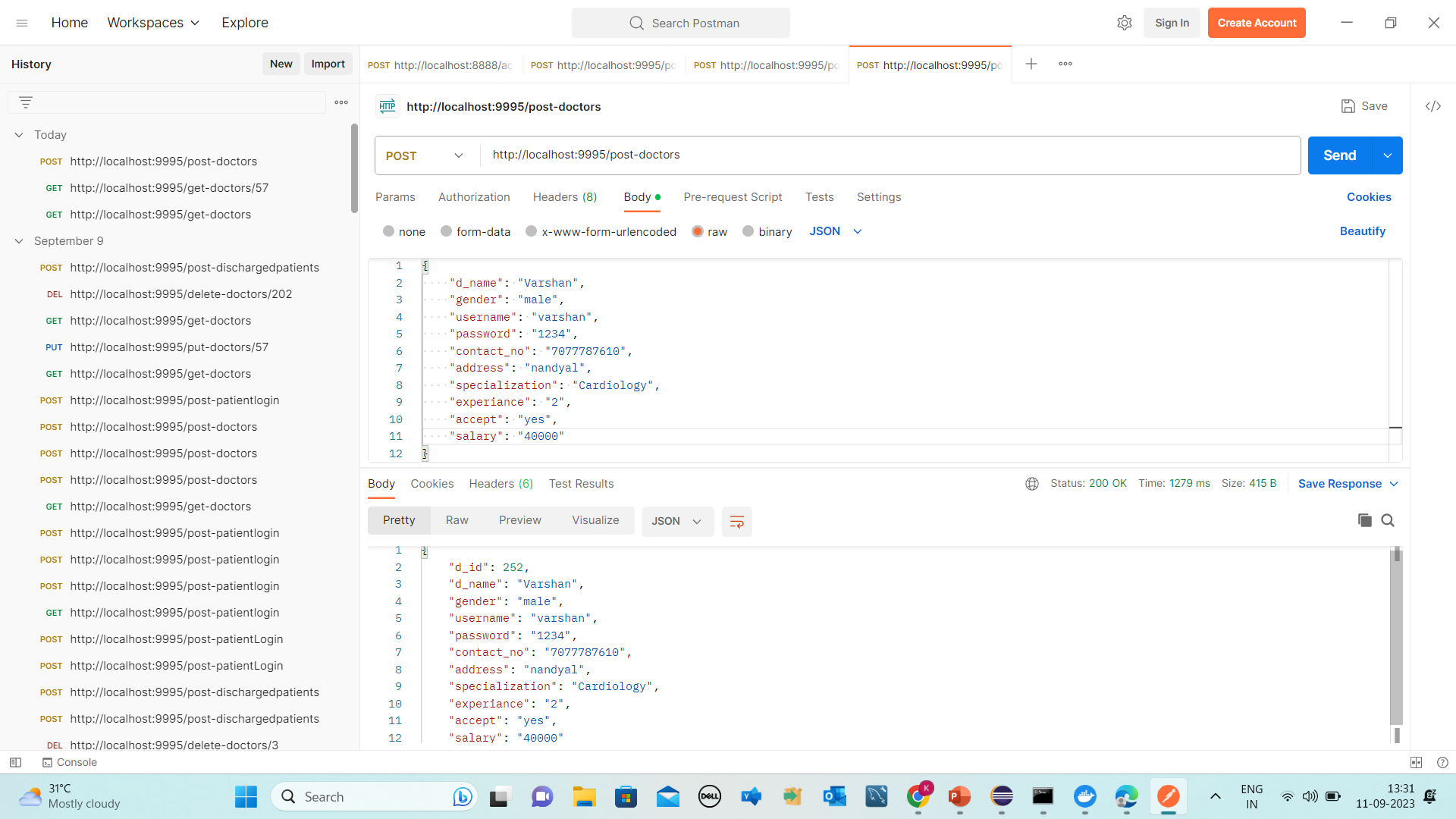
****

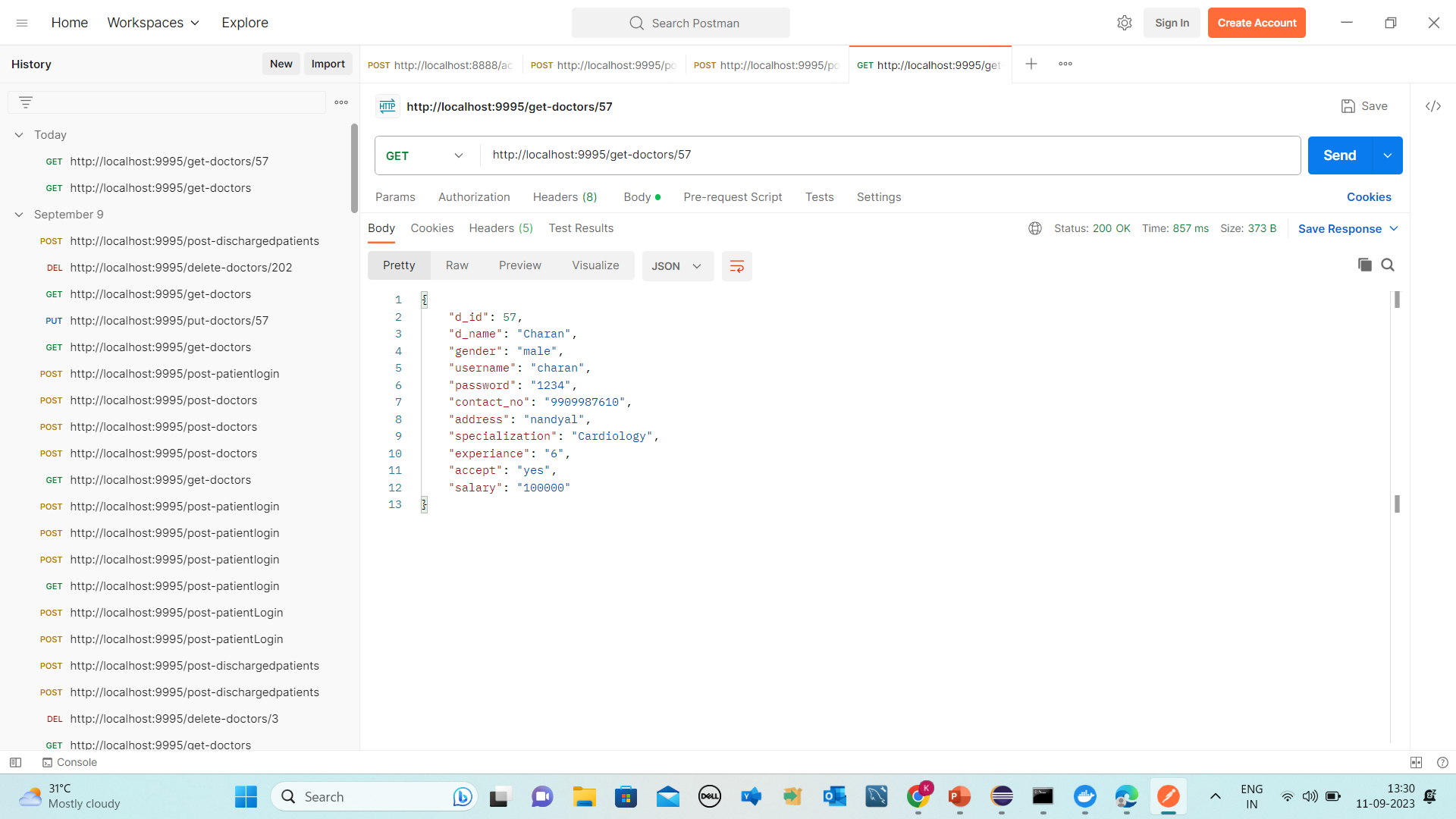
****

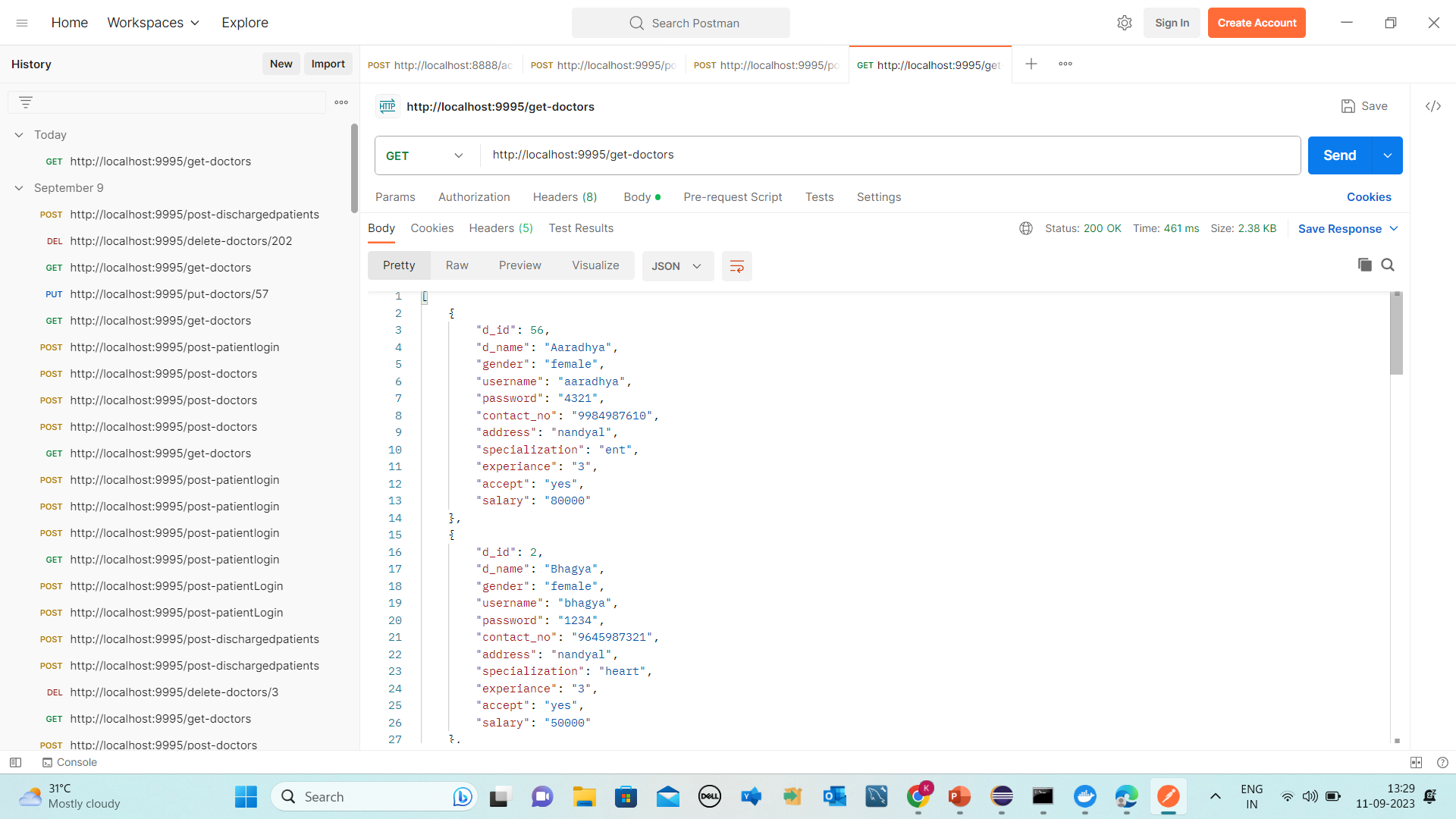
****

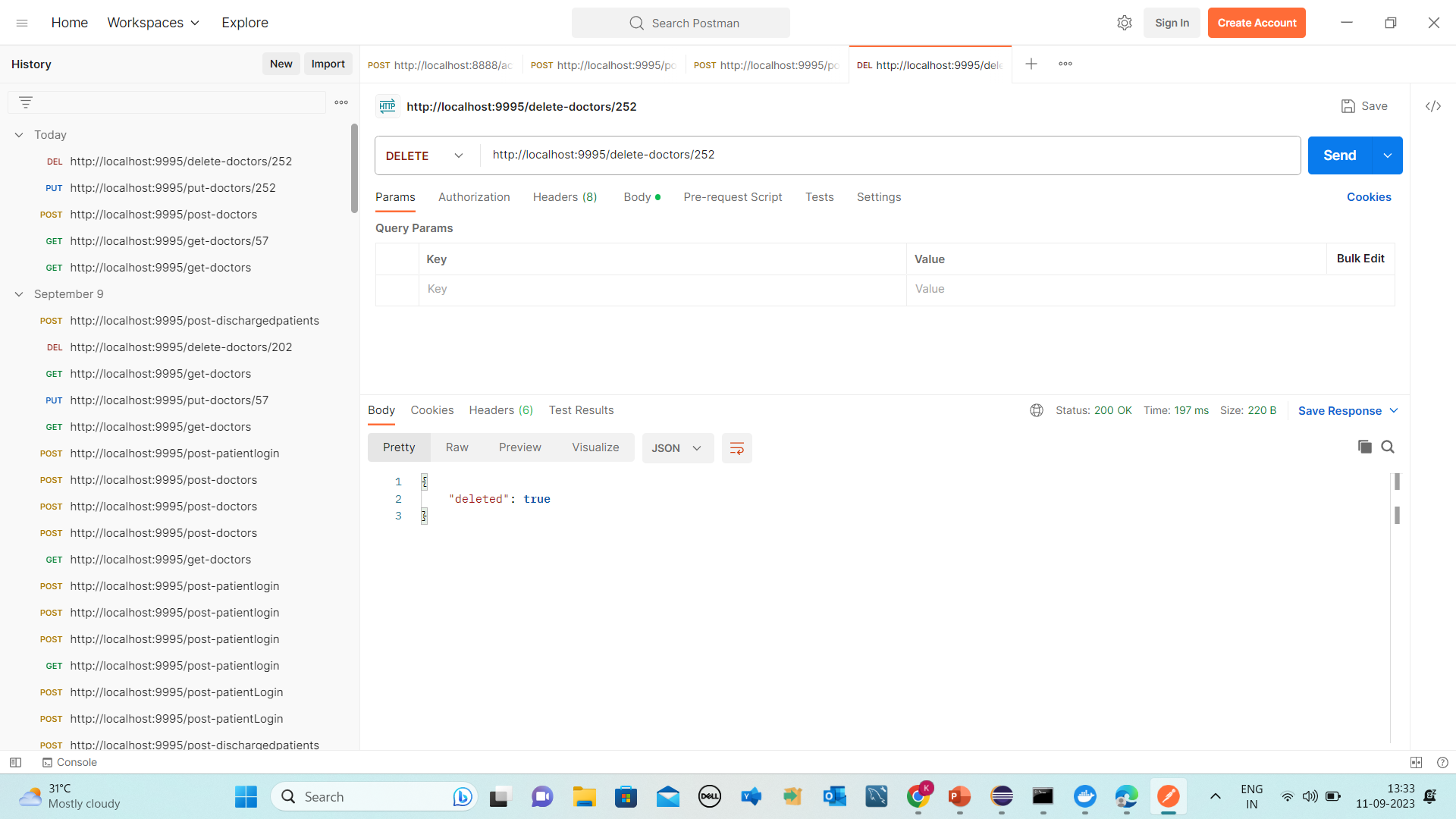
**Doctor-signup Screenshots:**

****

****

****

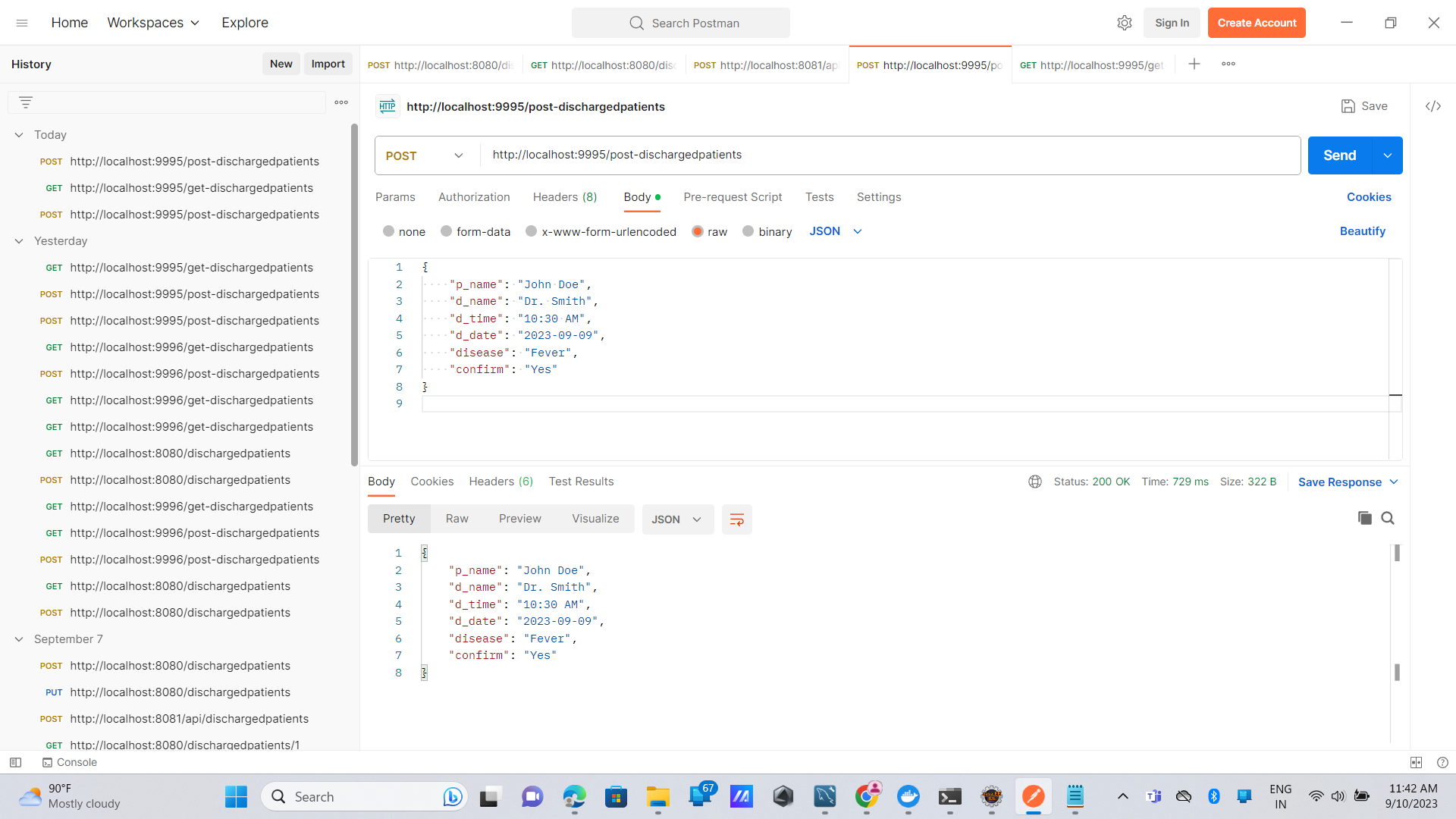
****

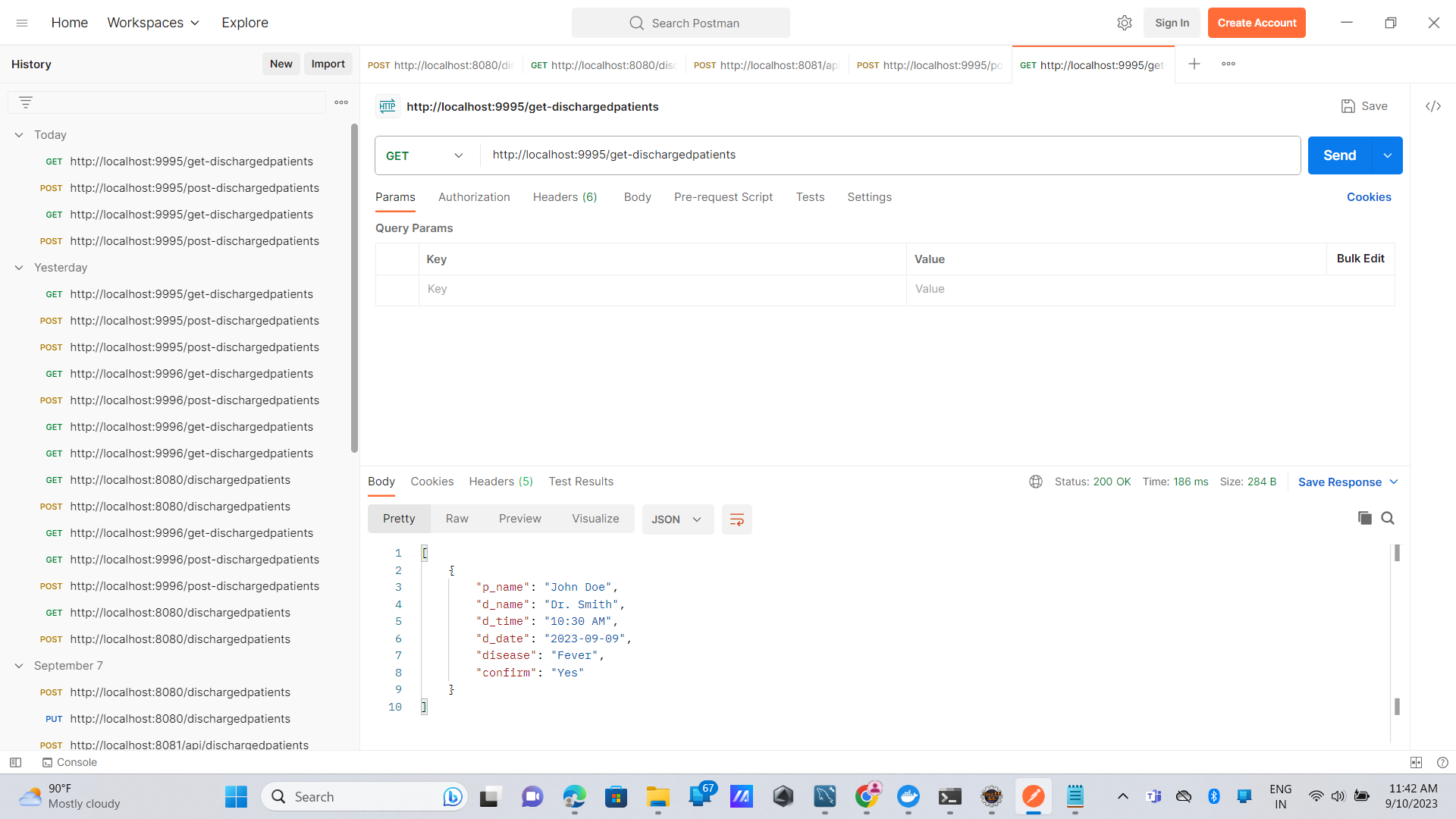
****

**Doctor Login Screenshots:**

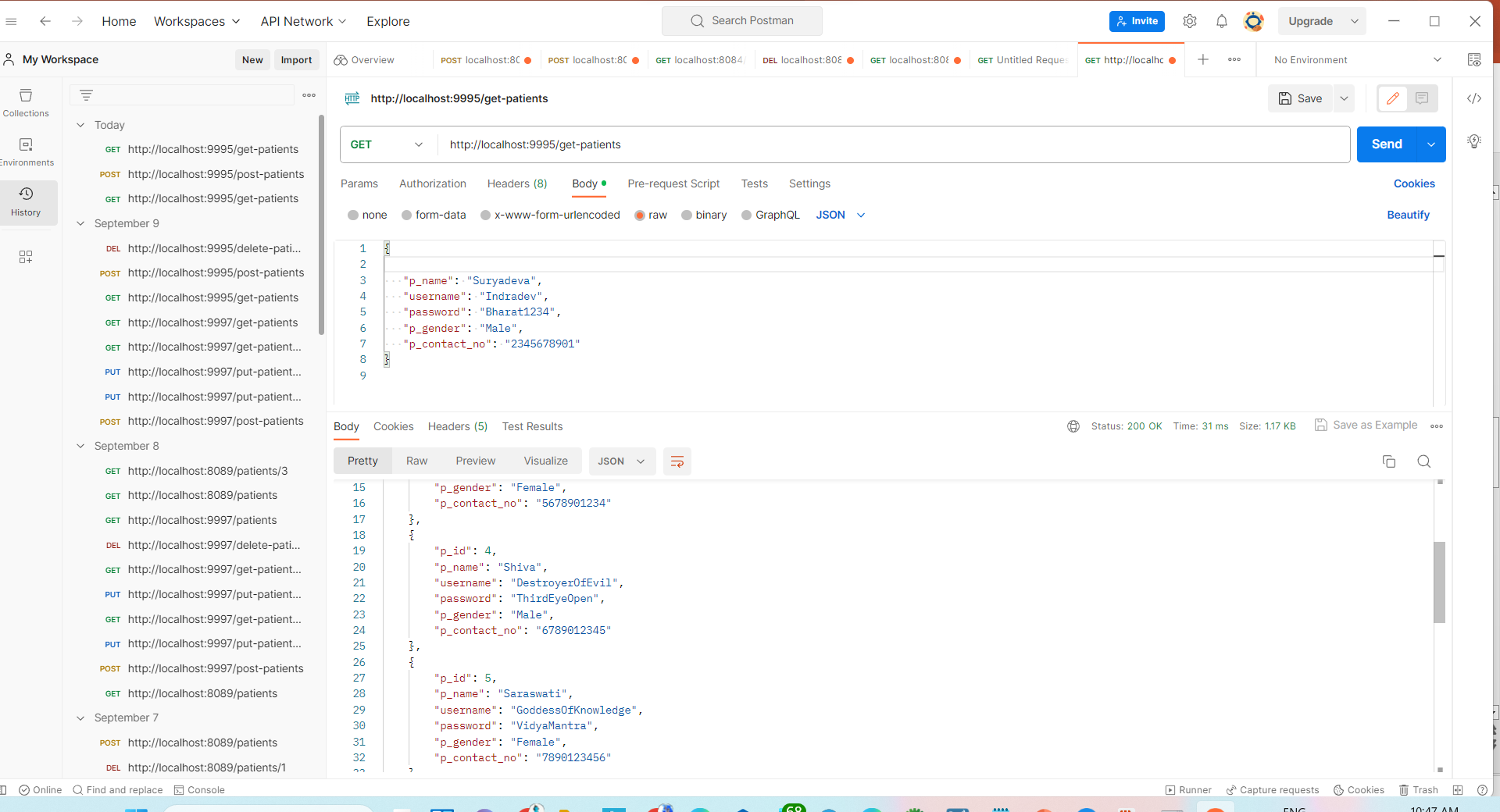
****

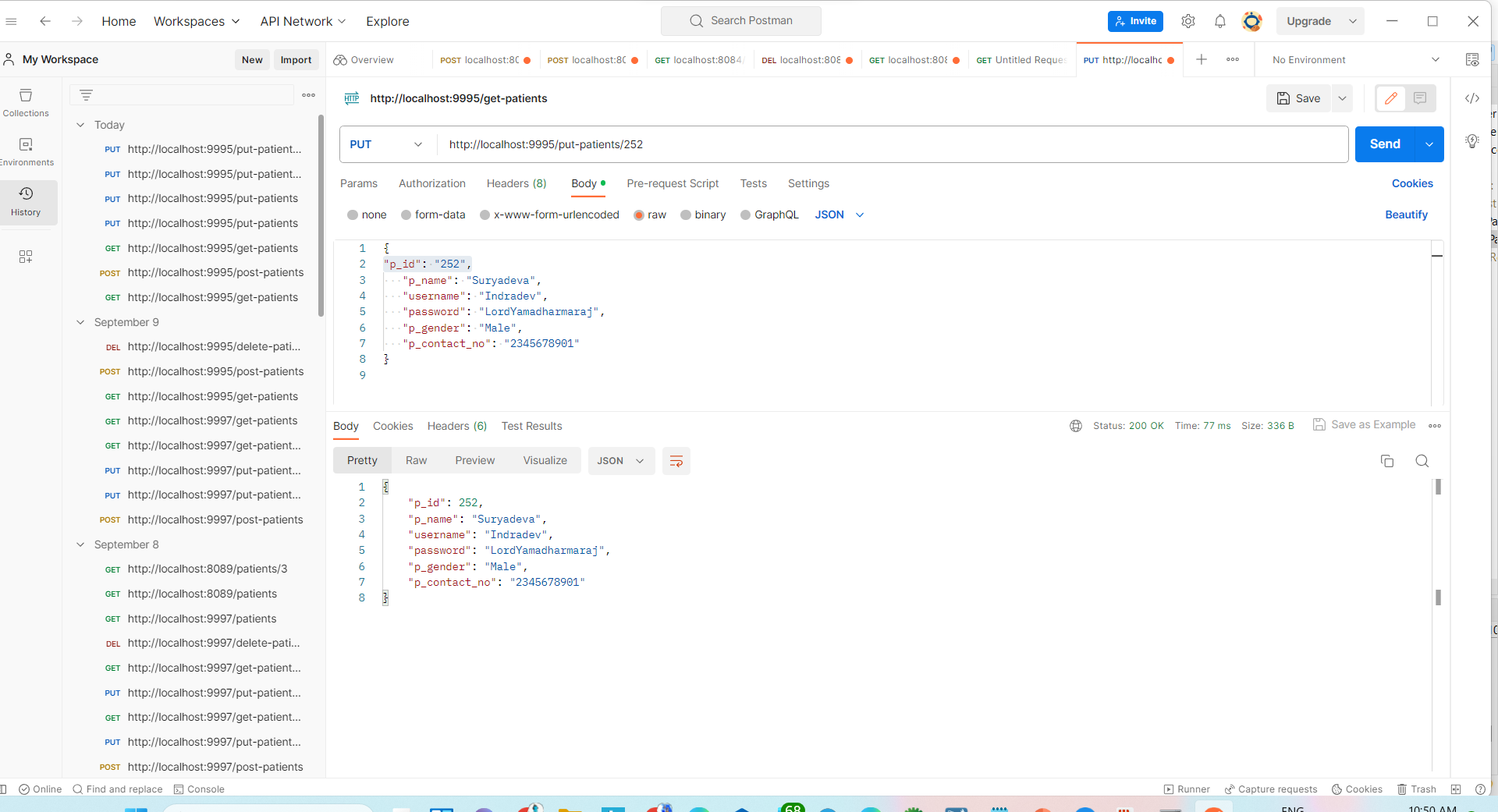
**Discharge Patients Screenshots:**

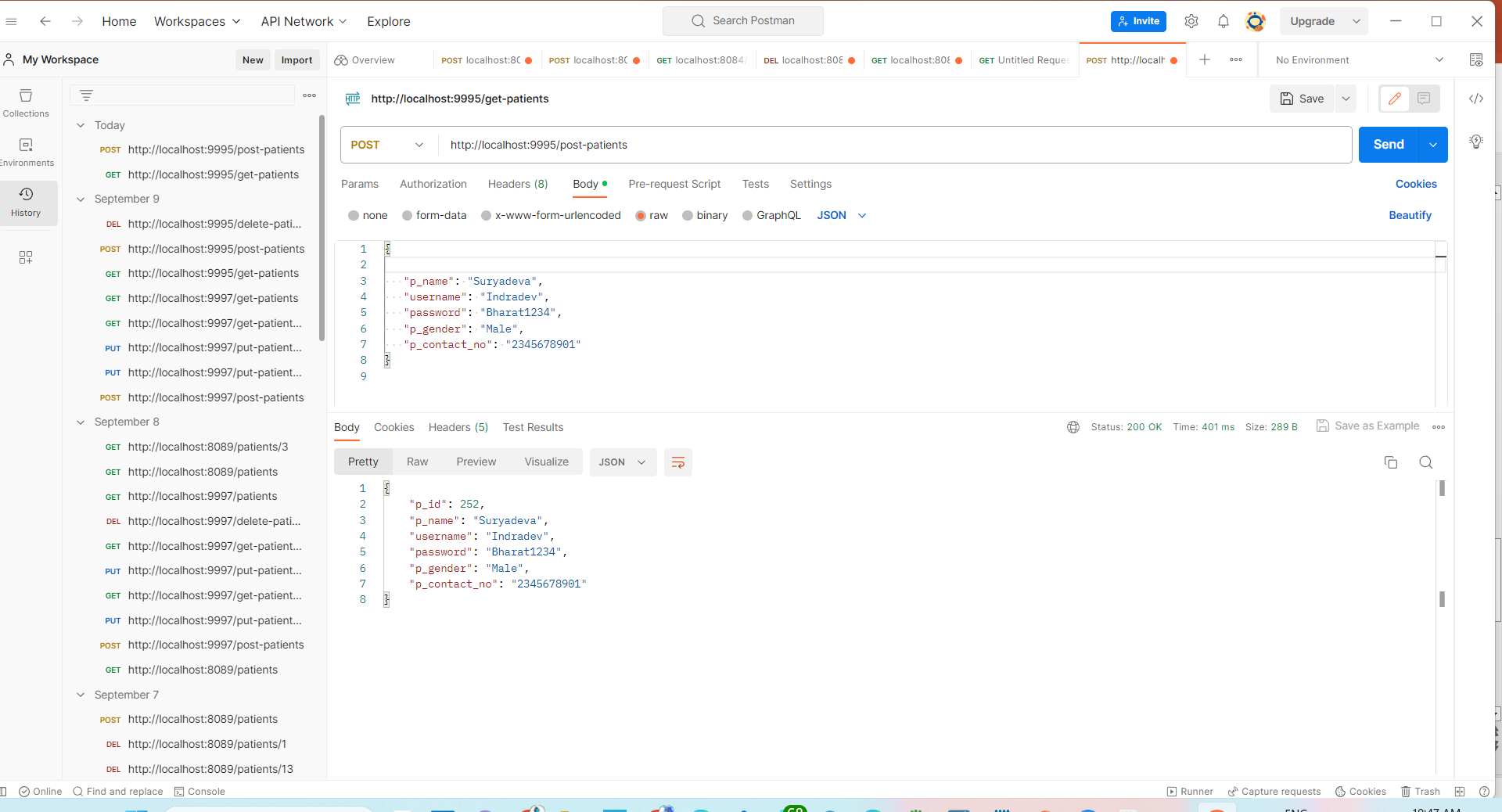
****

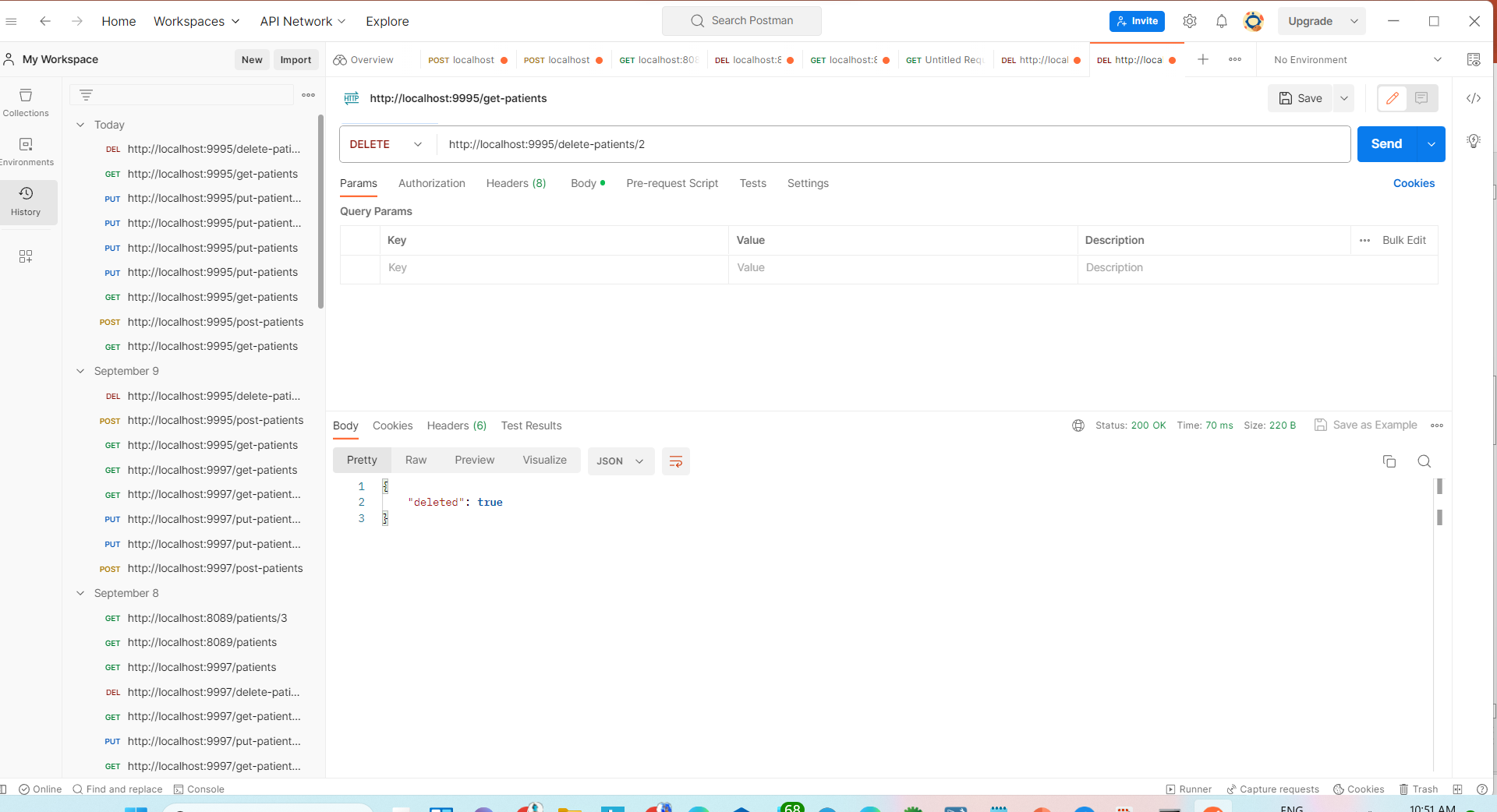
****

**Patient Sign up Screenshots:**

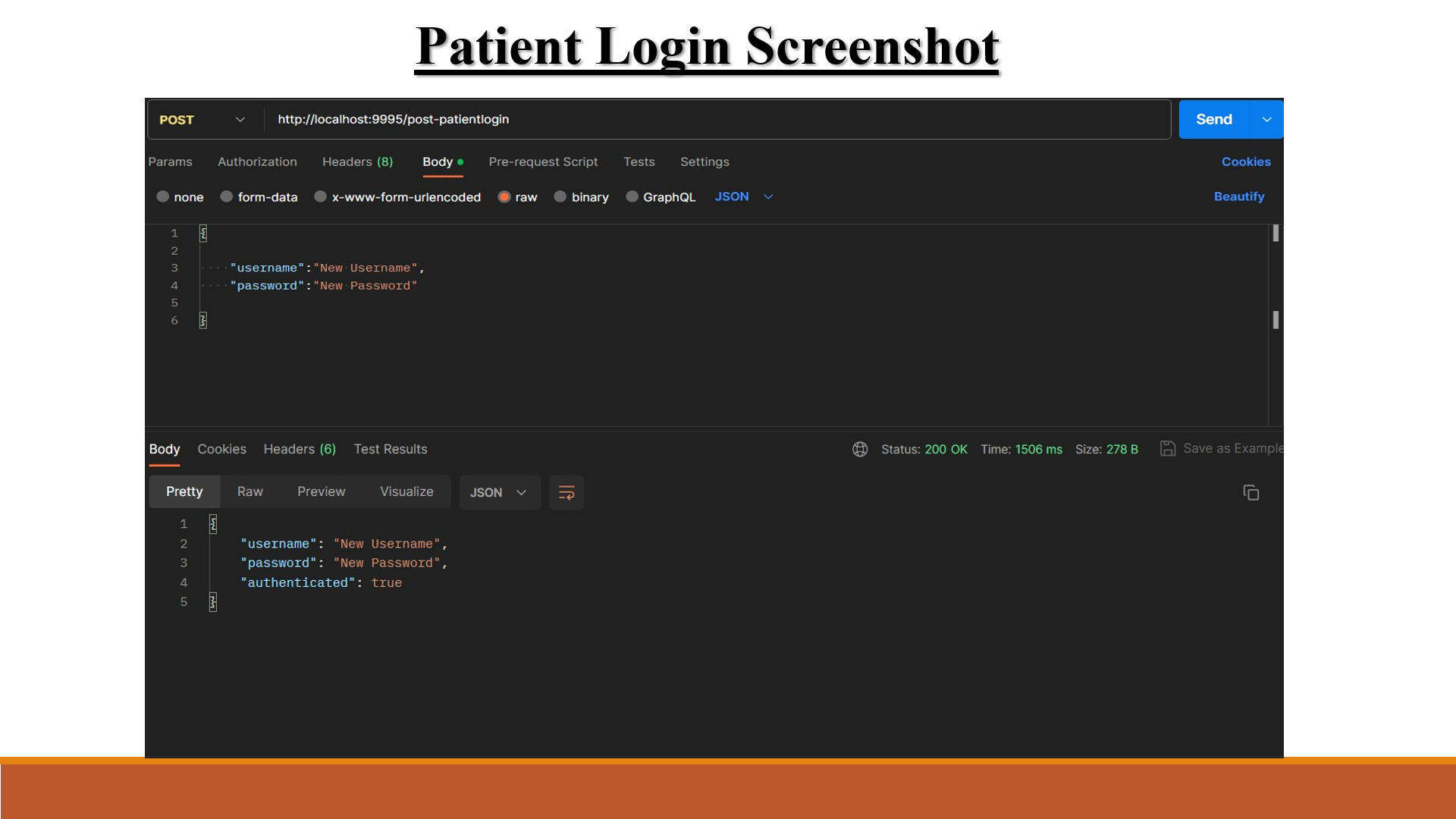
****

****

****

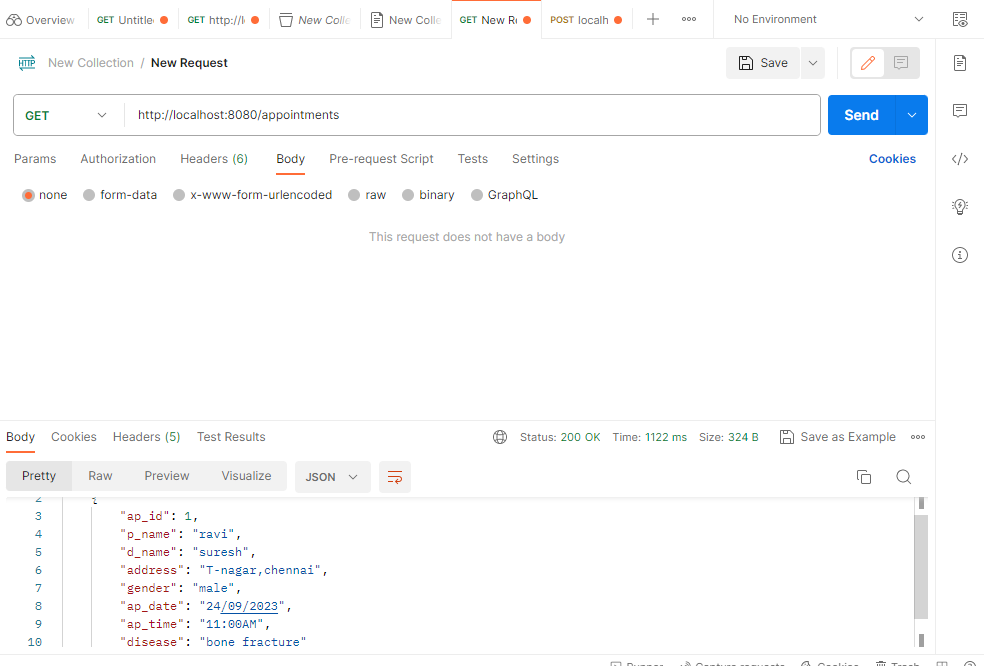
****

**Patient Login Screenshots:**

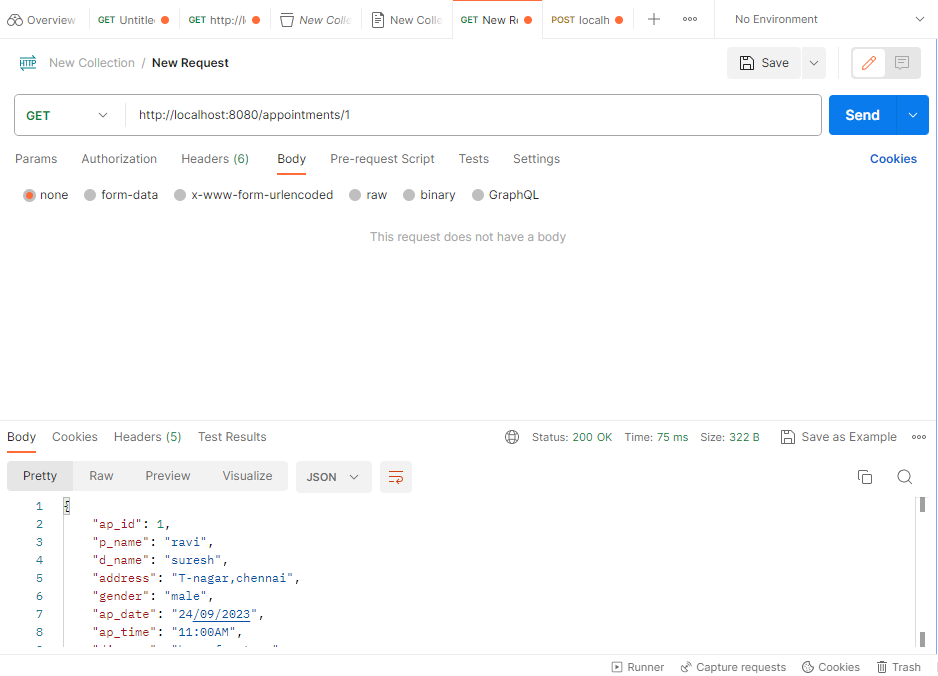
****

**Appointments Screenshots:**

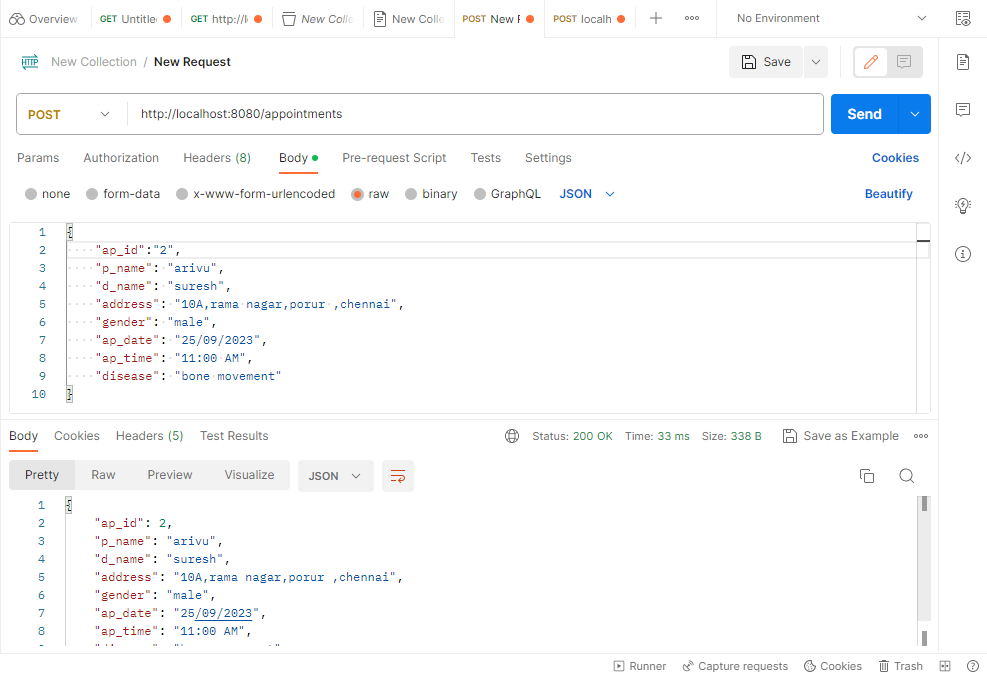
**1.Get all appointments:**

****

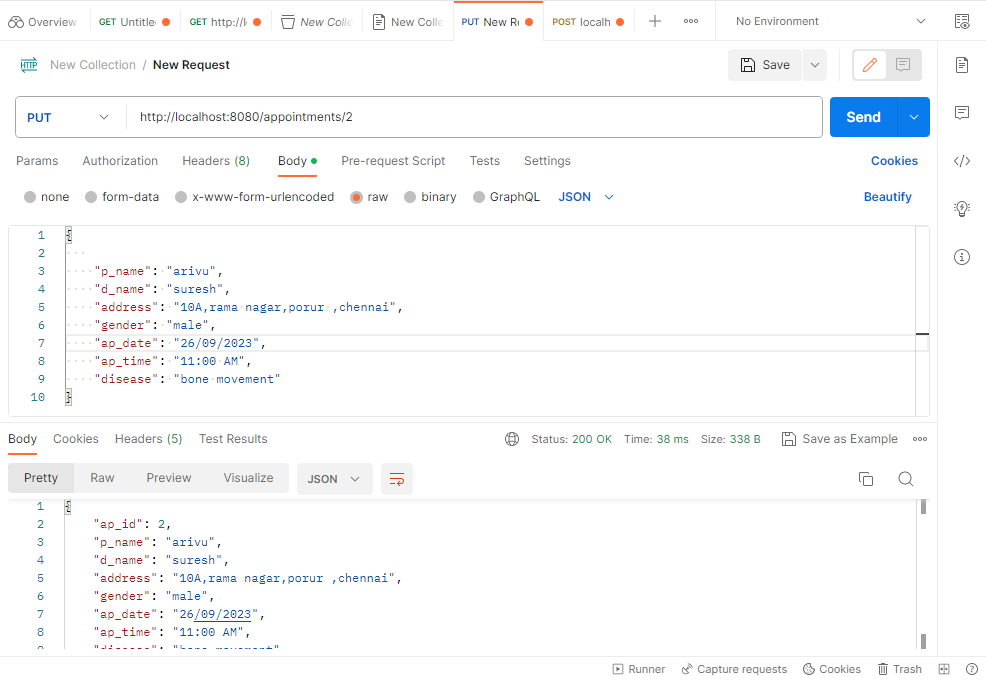
**2. get appointments by id:**

****

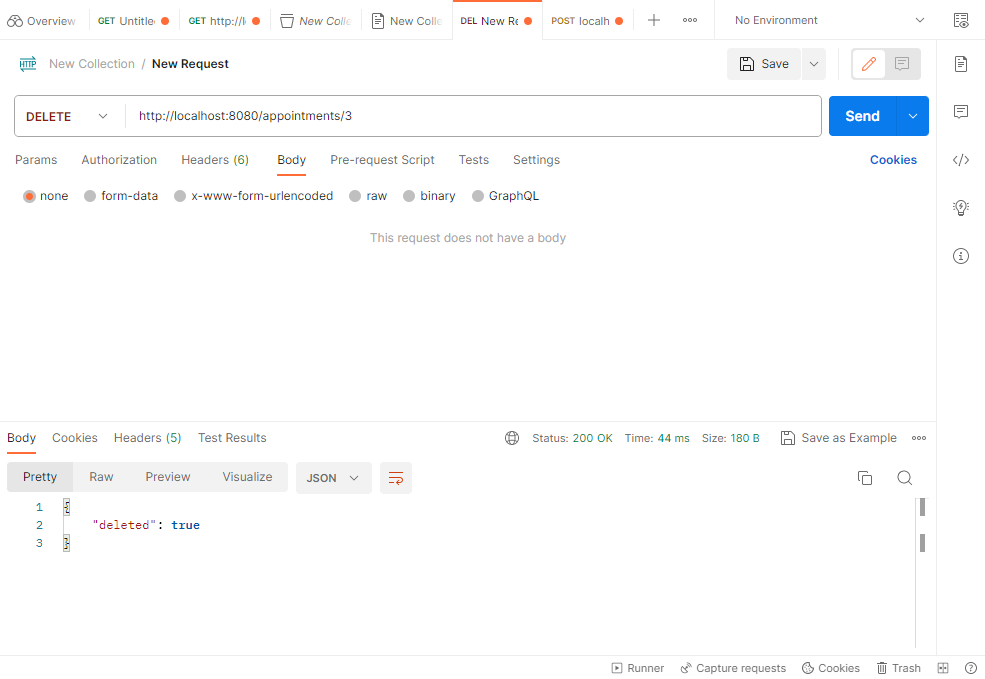
**3. post appointments:**

****

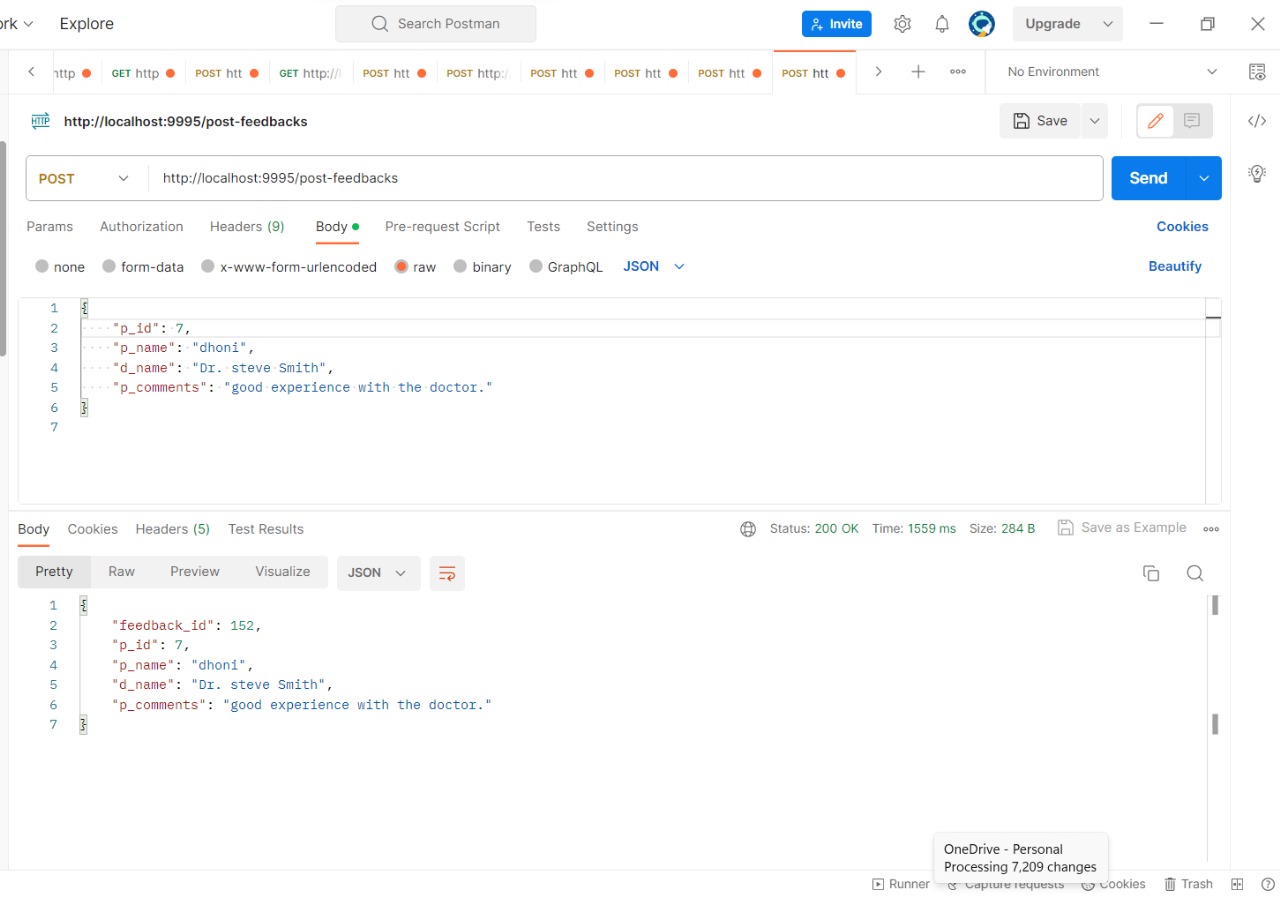
**4.update appointments:**

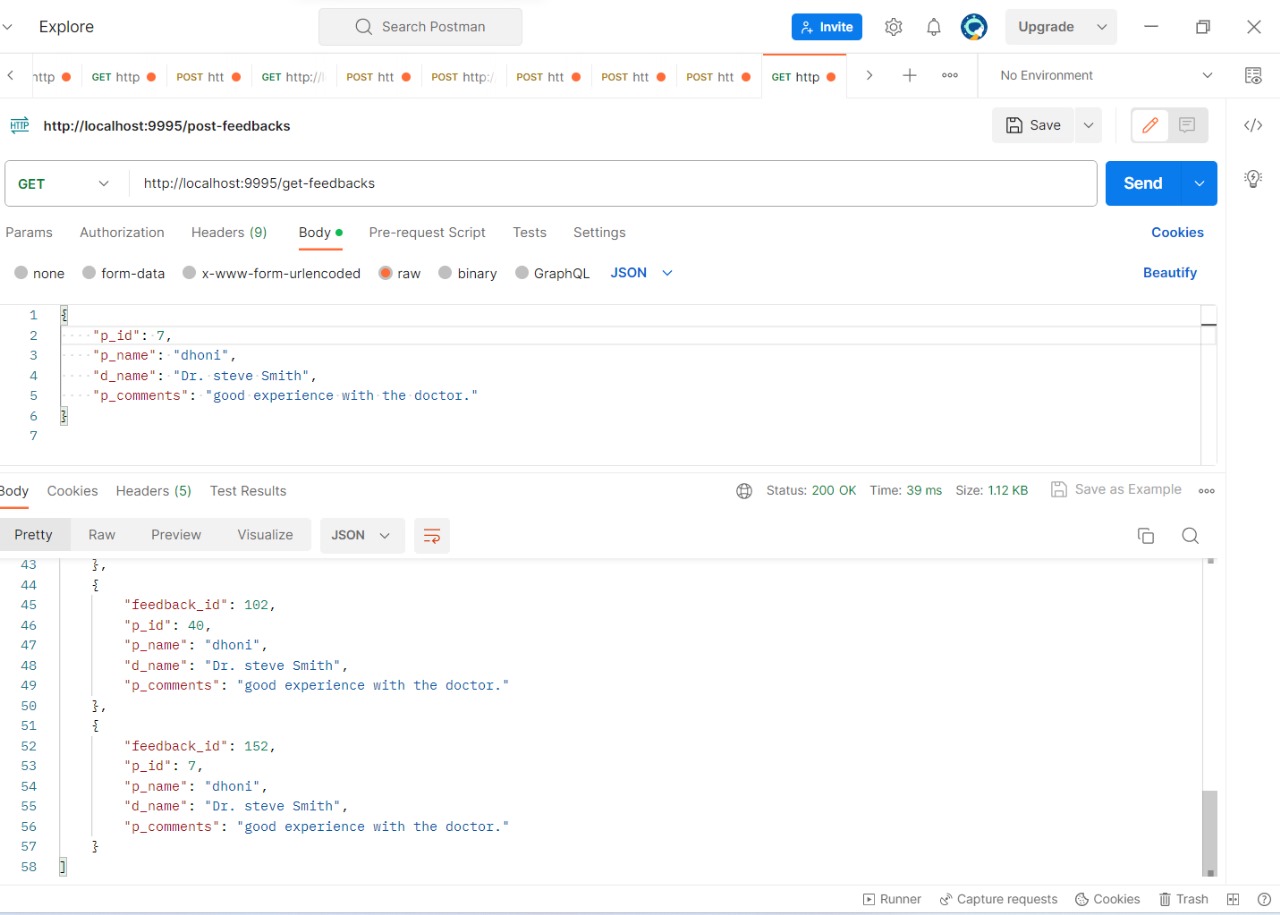
****

**5.Delete appointments by id:**

****

**Feedback Screenshots:**

****

****