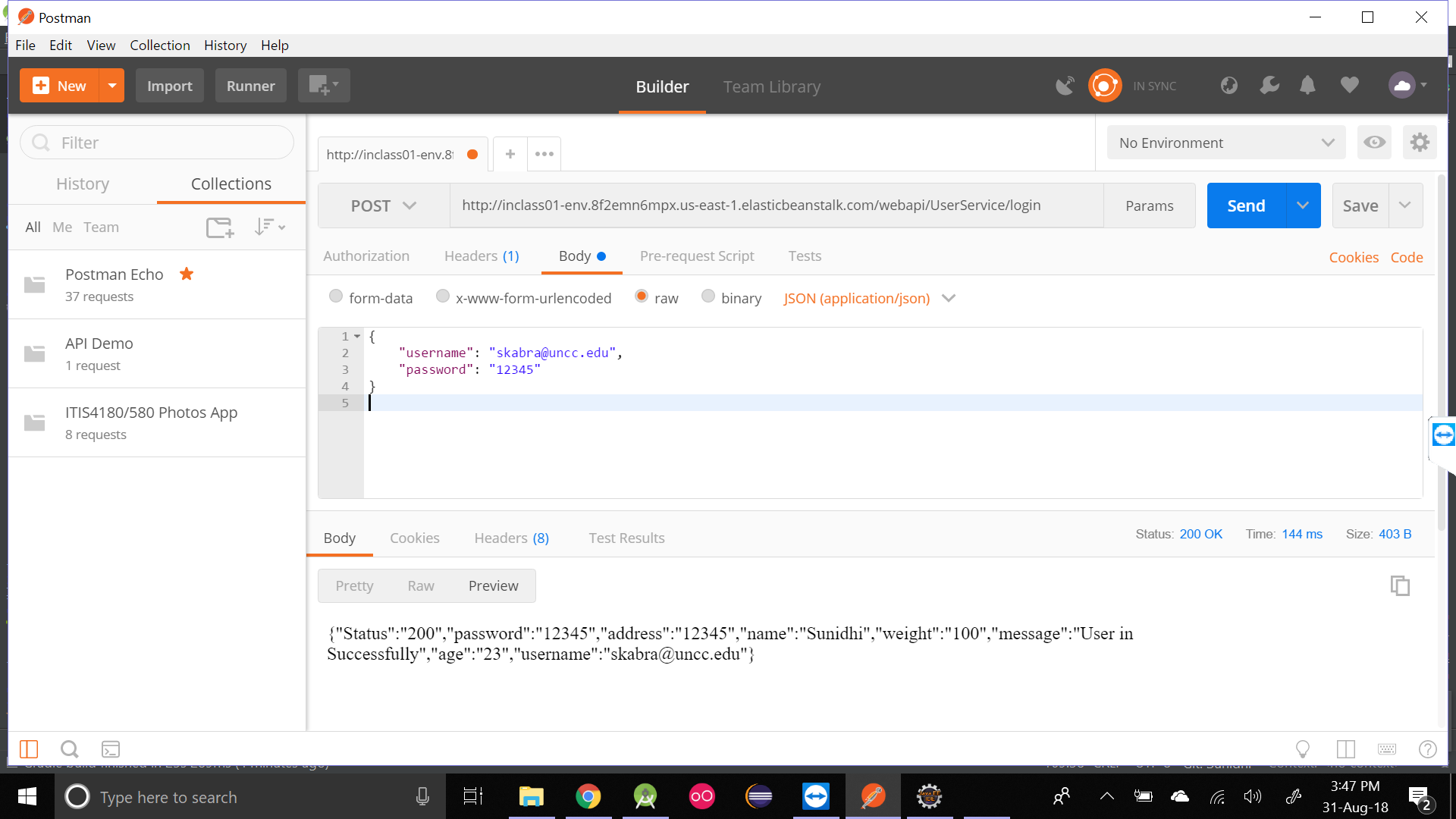
**IN CLASS ASSIGNMENT 1**

1. **API DESIGN:**
   1. The home page of the application has the features to login or sign up. The user enters the username and password and clicks login. The username and password are parsed as JSON string and sent in the JSON body.

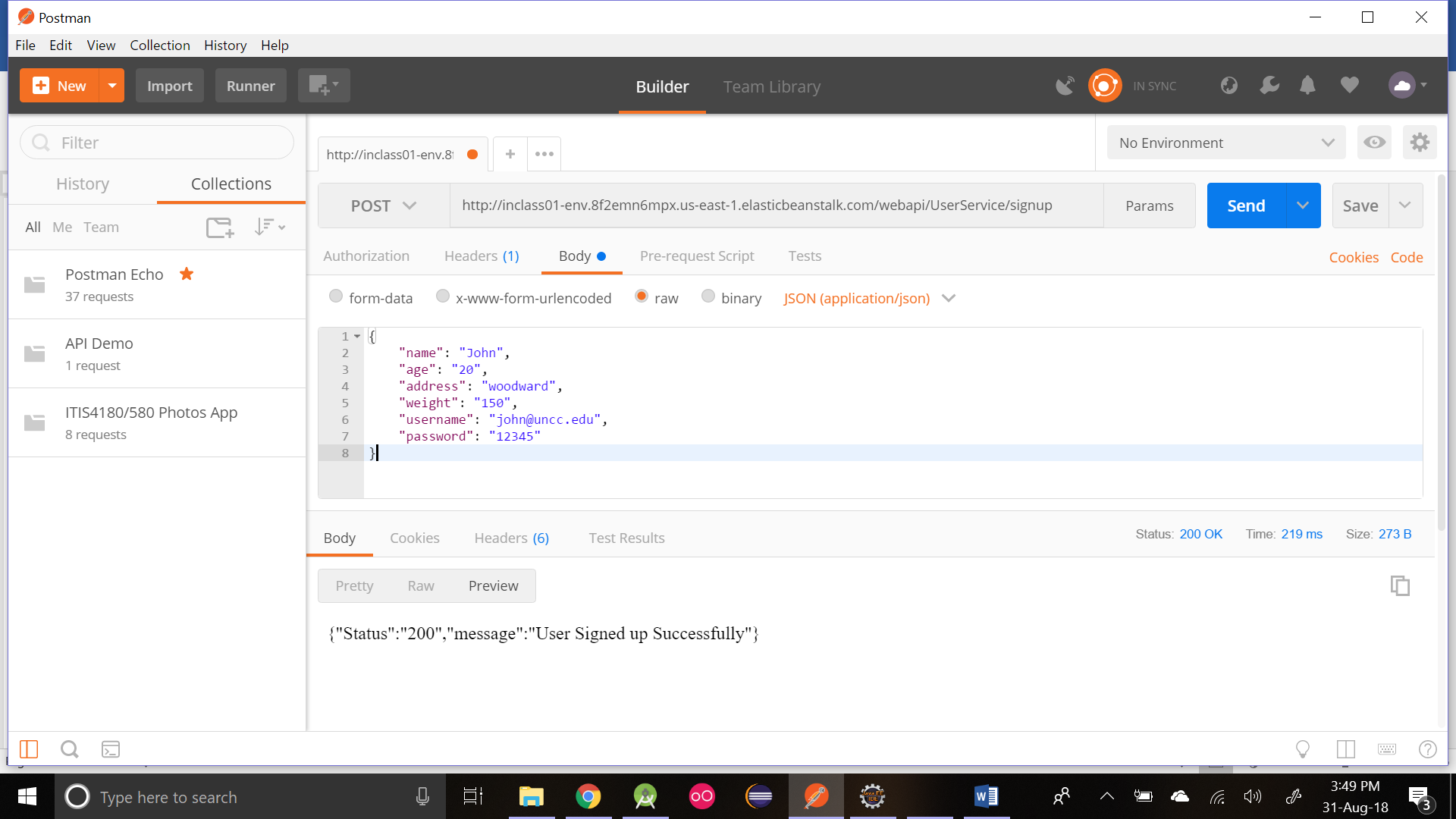
When the user clicks on Sign Up, the registration page opens. It takes the user’s name, age, address, weight, username and password. These details are parsed as JSON string and sent in the JSON body.

* 1. We used an Amazon RDS instance for remote Database connection and an Amazon EC2 instance for posting the API.

1. **IMPLEMENTATION:**
   1. The API is created using JAVA and the JAVAX Jersey Framework.
2. **API ROUTES:**
   1. API routes - google this - you will understand - this is basically how the data is transferred to what all platforms and what all places. For example - from our native android app the data entered in parsed in JSON string, included in the header of the API request and sent to the DB hosted on the AWS server. Once the query is successful, depending upon the type of the query, the API results a JSON along with a status 200. You will have to run everything on eclipse and show a demo of these things - include screenshots.
   2. Login: The user enters the username and password which is parsed in JSON string and sent in the JSON body to the RDS instance we created. If the query is successful, the API returns status (successful), message (User In Successfully) and user information.



* 1. Sign Up: The user enters the name, age, address, weight, username and password which are parsed in JSON string in the JSON body to the RDS instance we created. If the query is successful, the API returns status (successful), message (User Signed Up Successfully) and user information (name, age, address, weight).



1. **ASSUMPTIONS ON AUTHENTICATION:**
   1. For a user to sign up, it is a must for him to enter all the details like name, age, address, weight, username and password. None of the fields can be left blank.

For a user to login, we match the username and password. We are not checking if username and password are unique. Therefore, we assume that every person is just registering for once and they should remember their credentials.

1. **DATABASE STRUCTURE:**
   1. As this is a basic API demo assignment, we don’t have any dependencies and stored everything in a single table. We could have used a Relational Database with some dependencies like storing username and password in a separate table from other user information. For this assignment, we used a single table with attributes like person\_id (primary key), name, age, weight, address, username and password.