**Important for the Exam**

**TestCase:**

* Test Summary
* Preconditions
* Steps
* Test Data
* Expected result
* Priority (Optional)
* Actual Result(Optional)
* Status (Optional)

**Bug Report Structure:**

* Title/Summary
* Description (Steps to reproduce)
* Expected Result
* Actual Result
* Environment
* Attachments
* Severity
* Priority
* Status (Optional)

TO PRACTICE TEST CASES:

Lesson 5: Beet Sprout and Mighty Beet - For requirements

Lesson 6: MIGHTY BEET (TEST CASES)

Lesson 8: Beet Seed: Bug Tracking System

Lesson 9: MIGHTY BEET

Lesson 12: Beet Seed & Sprout:

Lesson 14: Beet Seed - Find 3 bugs and record them

Lesson 15: Beet Seed

Lesson 16: Beet Seed & Beet Sprout

Lesson 17: Beet Seed

**Explain the definition of validation and verification in your own words.**

Verification is the process of checking if a specific product is built according to its requirements or not. Essentially, it aims to answer the questions “is the product being built correctly and aligned with its requirements?”. It ensures that the software is correctly implemented according to the specific function it was designed to do.

Validation is the process of checking if a software product is being built according to the needs and expectations of our end user. It aims to answer the question “are we building the right product”?

* **Verification: Are we making the product right?**
* **Validation: Are we making the right product?**

**Give examples of unsuccessful product validation or verification that you have encountered in your life.**

* **Failed Validation:** Bank App for my credit card has a section with the advertising banners that are showing promotional campaigns. Banners are not clickable for the users, so we can’t get any information on those specific campaigns. The user is expecting to click and get more information, but it is not possible to do so.
* **Failed Verification:** A situation where that banner is supposed to be clickable but the development team didn’t build it accordingly.

**Explain the importance of 2-3 testing principles of your choice. Give examples from your own experience.**

1. **Testing shows the presence of defects, and not its absence:** Testing can indicate the presence of defects, but it cannot prove that there is nothing to fix. It merely reduces the probability of defects in a system/software.
2. **Exhaustive testing is impossible:** Running a complete testing with all the possibilities of input and output is physically impossible. Priorities in testing need to be considered.
3. **Early testing saves time and money:** Testing early in the software development lifecycle saves time and money. The earlier a defect is found, the cheaper it is to fix it.
4. **Defects cluster together:** Usually, small groups of modules contain the majority of defects and are responsible for most of the operational failures.
5. **Beware of the Pesticide Paradox:** If the same test is runned over and over again, eventually it will stop finding defects.
6. **Testing is context dependent:** Testing needs to be performed in different ways depending on the context. A production management software, where security is more critical, needs to be tested differently than an ecommerce
7. **Absence of errors is a fallacy:** It is possible that a thorough testing for all the requirements and all the defects found can still lead to a product that is difficult to use and does not meet the expectations of users.

**Define the concept of positive and negative testing**

* **Positive Testing:** Testing with valid values. It’s used to check the correct execution of system/ software functions.
* **Negative Testing:** Testing with obvious invalid values. Used to check exceptional situations to test incorrect uses of the system/software, but still probable to happen.

**What makes a good tester?**

Understanding the business value of the product.

**Make a comparative table of the most common methodologies:**

| Waterfall Model | Involves a sequential transitioning from one stage to another stage without skipping or returning to the previous stage. | 1) Easy and clear to understand.  2) Easier to manage because each phase has specific deliverables.  3) Requires highly qualified teams.  4) Best suited for the Aerospace Industry where you have unlimited time planning and budget. | 1) Almost impossible to go back to correct something that was not contemplated.  2) Not suited for industries where requirements are always changing. |
| --- | --- | --- | --- |
| V-Model Model | At each stage, the current process is controlled to make sure that it is possible to move to the next level. It is also a sequential path and each phase must be completed before the next phase begins. The testing of the product is planned in parallel with the corresponding phase of development, so testing begins at the stage of writing requirements. | 1) Also simple and easy to use;  2) Some testing starts before developing the code which could save time.  3) Defects are found at an early stage  4) Works well for small projects where requirements are easy to understand and fixed. | 1) Not very flexible when compared to other approaches.  2) Difficult to make changes later in the process  3) Requires a significant amount of planning  4) Requirements need to be well defined. |
| Iterative Model | Divides the project into parts (stages and iterations) and goes through the stages of the life cycle with every part. Each stage is independently complete, and the set of stages make for the final result. So we don’t need the requirements all defined. | 1) Improve the product step by step  2) Track defects at an early stage | 3) The fact that not all requirements are gathered up since the beginning of the lifecycle could lead to some system issues |
| Incremental Model | Based on the principles for the expansion of capabilities and improvement of modules and functions of the program. This “increased by one” approach is used to indicate product versions. After each iteration there is a new version of the product | 1) Generates a working software easily and quickly  2) Lower initial cost to deliver a MVP  3) Easier test and debug during each iteration  4) Feedback from the customer | 1) Needs a clear and complete definition of the whole system |
| Spiral Model | There are successive iterations. During the first stages, product specifications are refined and then new features and functions are added. | 1) Risks are mitigated after each iteration | 1) Costly to implement |
| Agile Model | Development happens via short cycles. At the end of each of them, the customer receives a working product. | 1) Little planning is enourhg to start the project  2) Higher customer satisfaction, there is continuous delivery  3) MVP ready in an early stage  4) Late changes in requirements are not a problem | 1) Difficult to assess the total effort required at the beginning of the development cycle  2) Projects can be taken off track if the final outcome is not clear |

**In your opinion, why did the Agile manifesto appear? And what problems did it have to solve and did it succeed?**

The agile manifesto was created as an alternative solution to the traditional and more rigid and inflexible software development approach that existed at the time. The approaches were known to be driven towards documentation, and with rigid and inflexible development processes. There was an increasing need to make the software development cycle more agile, with more flexible and adaptable processes, but still following a structure.

It helped solving problems like:

* Inflexible and rigid software development processes: Traditional processes are more rigid and slow. Have more difficulty to handle changes in requirements or unpredictable challenges during the development phase.
* Long development cycles: We needed to wait until the end of a life cycle to have a working software product, and have stakeholders feedback on it.
* There was no customer or involvement of the user. So a higher risk that it didn’t meet their expectations.
* Slow when it comes to adapting and releasing new features that follow trends and market changes

**You are the founder of a startup planning to launch a mobile application for sharing cat photos.**

**What methodology will you choose for the development process and why?**

Agile methodology: allows me to build an initial App with basic features, and have an MVP that I can quickly launch into the market and then refine it according to customer feedback and needs. It helps me improve and prioritize features that users like or missed, and it allows me to have a return on investment early. Also, I would opt-in for the Scrum framework that would allow me, and my team to have a clear real-time view of the project.

**Give short examples of requirements (3-5) for any item from your environment that would meet each of the following quality assessment criteria:**

* **Atomic:** Should contain only a single element to test

The user logs in in the App using a face ID

The user takes a photo by clicking the button “Take a Photo” on left bottom corner

The user makes a purchase by clicking on the button “Purchase Now”

The user can make comments with a maximum of 90 characters

* **Consistent:** Requirements don’t contradict other requirements

User comments have Arial font and font size of 12

User is able to download the App in either IOS or Android operating systems

Dates that reference the submission of a photo are displayed in the format "MM/DD/YYYY"

A user always receives an alert, inside the App, when someone comments on the photos.

* **Testable:** Requirements should be able to be tested

The user logs in into the APP in 10 seconds or less

The user uploads a maximum of three photos each time when clicking on the button “Upload” on the top corner

The receive an alert notification, inside the App, when someone comments on their photos

* **Traceable:** Requirements need to be traced between business requirements, technical requirements and test cases ID

The user logs in in the App using a face ID - **B1, T92, Test Case ID 1**

The user takes a photo by clicking the button “Take a Photo” on left bottom corner **B2, T94, Test Case ID 2**

The user makes a purchase by clicking on the button “Purchase Now” **B3, T98, Test Case ID 3**

The user can make comments with a maximum of 90 characters - **B4, T96, Test Case ID 4**

**In your opinion, which of the requirements testing techniques guarantees the highest possible quality of the final result? Support your answer (3-5 sentences).**

The best way is via “Test Cases and Checklists”. We can assess if we have high quality requirements that will support the development of our software, and ensure that they are clear, understandable and align with the product purpose.

Using checklists to match requirements with test cases, helps us facilitate the verification process and measure the quality and relevance of the requirements.

**You are the founder of a startup planning to launch a mobile application for sharing cat photos.**

**Come up with functional (5-7) and non-functional (also 5-7) requirements for the application.**

**Functional Requirements:**

* The user logs in into the App using an email address and password
* The user is able to upload up a maximum of four photos each time
* The user can upload photos by clicking on the “Upload” button
* The user can publish a comment by clicking on the “Publish” button in the description field
* The user can only upload cat photos
* The user can make comments with a maximum of 90 characters

**Non-functional requirements:**

* The uses can download the APP for both IOS or Android operating systems
* The user can upload a photo in 1 minute or less
* The user can download the App in 5 minutes or less
* The App is available in both English and Portuguese
* The company logo needs to appear on the top right corner of the App

**Make a comparative table of the three types of test documentation:**

| Checklist | List of checks that must be performed during the system testing. It contains only the name of the tests that need to be performed. | 1) Quick to build: better for time efficiency  2) Simple to create and to use | 1) Lack of detail when comparing test cases, so it can be less effective in complex projects. |
| --- | --- | --- | --- |
| Test Cases | Test Case in a test artifact that describes the steps, conditions and parameters required to test the implementation of a function or its parts. | 1) More structured and systematized  2) Easier to track testing efforts, since we have detailed steps, conditions and expected outcomes. | 1) Requires a significant amount of time to built it |
| Use Case | Detailed description of how a system interacts with a user. If focus on the user needs, and the desired outcome. | 1) Promotes the user needs and expectations  2) Gives the team a better understanding and guidance of the system behavior from the user perspective | 1) Overlooks non-functional requirements |

FYI: User Story: Informal narrative that describes a specific feature from the perspective of the end user. If focus on the user needs and the desired outcome of an action in his perspective.

**Make a checklist for testing the main page of your favorite online store (https://www.asos.com/)**

* Asos site opens
* Search bar functionality
* Insert User Credentials in User Profile
* Browse Product Categories in Women section
* Browse Product Categories in Men section
* Add Product to Cart
* Product Purchase
* Social Media Functionality works
* Newsletter Sign-up works
* Currency selection

**Create 5 positive and 5 negative test cases to check the basic functionality of your favorite online store. (https://www.asos.com/)**

<https://docs.google.com/spreadsheets/d/146EnD8_OPfxxXDZcDhIMh7kbk2xF1JGoVV8iO_1T8V8/edit#gid=1198009978>

**Make a comparative table of functional, non-functional, and change-related types of testing.**

**The comparison should contain the following blocks:**

* **what is being checked;**
* **when applicable;**
* **restrictions;**
* **peculiarities.**

| **Functional Testing** | What is? | When we use it? |
| --- | --- | --- |
| Functional Testing | - We use it to test if a feature or a function is doing what is expected to do.  Essentially, is the output of a specific action the expected results | - Both during software development or after doing changes to the software (it can be to launch a new product, do a fix, or new features).  - Applicable in all testing levels (unit, integration, system and Acceptance) |
| GUI Testing | - We use it to test the interface according to the requirements (size, font color, consistent behavior). | - Right after development implements the software interface |
| Security Testing | The security of the system (risks from hacker attacks, viruses, access to confidential information) | - Before launching the software into production |
| Interoperability | - Ability of the application to communicate with other systems or components. Covers compatibility testing | - During the integration testing level phase |

| **Non-Functional Testing** | What is? | When we use it? |
| --- | --- | --- |
| Performance Testing | - How the system performs under a particular workload. There are different types of tests that can be performed and help testing reliability and scalability of the software | - Close to launching the software product  - During the Acceptance testing level phase |
| Installation Testing | - Test if the installation/configuration is successful. It also covers testing updates or installation software | - Close to launching the software product  - During the Acceptance testing level phase |
| Usability Testing | - Test to confirm the level of convenience, clarity and user experience | -During acceptance level phase  - Close to launching the product |
| Failover and Recovery Testing | -Test if the product can recover from failures, faulty hardware or other problems | -System Level phase |
| Configuration Testing | -Can a software system operate under different system configurations (operating system network configurations, etc.) | -During system acceptance level phase  -Right before releasing a product |

| **Non-Functional Testing** | What is? | When we use it? |
| --- | --- | --- |
| Smoke Testing | Short cycles of tests that confirm the software is still performing the main functions after changes | When changes to the code happen (launch new features or bug fixes) |
| Regression Testing | After changes in the software code, aim to check that all the functions of the system are still working. This is a broader test aimed to validate the entire software system, so it can be very time consuming. | Everytime there is a considerable update to the code (improvements, new features). |
| Re-test | Confirm that a defect has been fixed by re-testing the same steps that identified the defect issue in the first place. | Whenever we fixe a defect in the code |
| Build Verification Testing | Test if a created build version meets the quality criteria and is stable before it is released for further testing | Always applicable when a new build is created |
| Sanity Testing | Is part of the regression testing, but focuses on specific functionalities, so it’s more narrow and quick. It verifies that recent code changes have not impacted specific functionalities. | Executed after change in the software code that might impact other functionalities. |

**Explain the difference between regression and retesting (5 sentences).**

Regression testing and resting are both change related types of testing but have different purposes. Re-test aims to confirm that a defect has been successfully taken care of, and functionality is working as expected by retesting the steps that lead to that defect. Regression testing is a much more exhaustive type of testing that aims to test if, after a code change, all the functions of the system are still working as expected. This is a broader test that aims to validate the entire software system

**Do you think it is possible to perform only functional testing for a product without checking non-functional requirements?**

It could technically be possible to perform only functional tests in very particular cases. Since functional tests ensure that the software meets the functional requirements, so doing what is expected to do, we can make sure that the software output is going to be what is expected. However, it could potentially build a software that is not usable and doesn’t meet the users needs, and we could also miss testing important features such as security and reliability of the systems.

**How do you see the need for smoke testing? Is it always appropriate?**

Smoke tests are always appropriate right after a new software build is made or after a new software is deployed. So it’s alway s appropriate when there is a software code update.

**In your opinion, what is the greatest benefit of using test management systems? Which of their features allow to achieve this benefit? Support your answer.**

Using a test management tool has a lot of benefits such as:

* **Saving Time:** Having a centralized tool with all test cases and test runs in one place helps the team saving team in terms of test status, and track progress. Also, they can just reuse test cases and use templates so it’s easier to write test cases and launch test runs.
* **Traceability:** Easier to trace back test cases, to user stories and to defects since everything will be centralized in one tool
* **Facilitates organization:**  The fact that we have all the test cases and test runs (with results, defects) in one centralized place, ensures that teams can easily access and manage their projects, and organize all of our testing information.
* **Scheduling and tracking:** A test management tool also helps us schedule upcoming and current projects better, features like Milestones help us keep track of when specific Test Runs need to happen, their testing coverage, and this way it’s easier to control the deliverable of specific projects and product launches.
* **Accesses:** it’s easy to give access and share the project instance with stakeholders within the company when necessary. Also, a test management tool allows tests to be assigned to specific people within the team, so it’s also easier to manage a team of testers through a test management tool.
* **Test Cases:** A test management tool also lets us copy and move test cases, and use test case templates that can save time during the process of creating test cases.
* **Live Dashboards:** Access to live graphs that give us testing coverage (percentage of passed versus failed) within test runs and milestones.

**Give your own examples of bugs that may have such combinations:**

Severity - Critical / Priority - Low

* Beta version of new feature: As a critical functional defect but doesn’t have many users that use it. For the tester it is critical severity but for the project manager it is low priority because not many users use it, so it can be fixed in the next release.
* The user doesn't purchase confirmation even though the purchase was successfully completed, and this was a requirement.
* Severity - Minor / Priority - Highest
  + Inconsistencies or Mispelling of the name of the brand or using an old logo version.
  + Inconsistency in the colors of buttons, or headers through the application

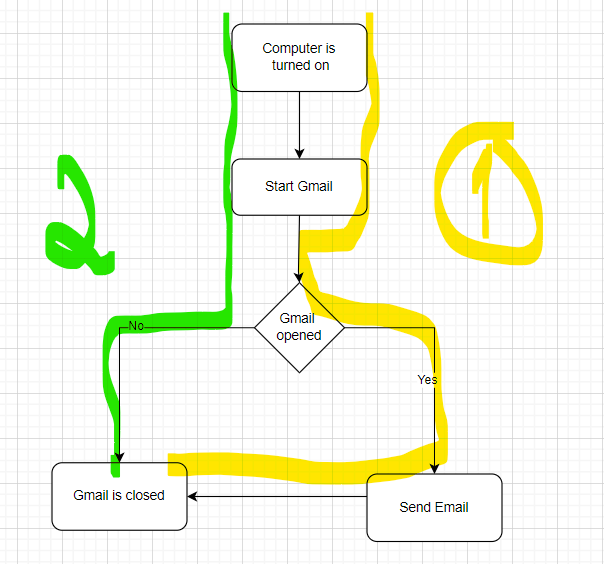
**Make a comparison of static and dynamic testing techniques. Give the advantages and possible limitations when using each of them.**

| **Static Testing** | **Dynamic Testing** |
| --- | --- |
| Static testing is a method performed to check defects on a code without actually running the software application. Is performed early on the software development cycle, with the purpose of avoiding defects in an early age, since it’s easier to fix defects at this stage.  It involves reviewing documentation, user stories, diagrams, instructions and other documents.  Also, you evaluate the quality of the code written by developers. It can involve running automated testing of the code and documentation to find obvious errors and bugs. | Dynamic Testing performed to analyze the dynamic behavior of the code while running it. So that dynamic testing can happen the code needs to be written, compiled and run.  This includes working with the software giving input values and checking the output is as expected, by executing test cases through manual or automated tests. |
| Ad1: Debugging is easier before moving to dynamic testing | Ad1: Covers issues related with software/system executions (runtime, memory usage, etc.), performance and and response time |
| Ad2: Good to catch bugs that are difficult to catch during dynamic testing | Ad2: Analysis features, and interactions in the various software components |
| Ad3: Prevents design and build the code without inconsistencies, so we have a higher quality code | Ad3: It helps cover a wide range of defects |
| Ad4: Reduces time and cost of the development and testing later on | Lim1: Time consuming and requires a high amount of resources |
| Ad5: Easier to maintain the code because of the early detection of defects | Lim2: Defects that are found in dynamic testing take a higher time to fix due to the challenge that could be tracking it’s origin |
| Lim1: We don’t run the software so we can’t find any bugs related to the behavior and performance of the software | Lim3: More difficult to identify the root of a defect |
| Lim2: Focus on identifying errors in documentation or flaws in logics, so it can miss issues with the interaction of the software | - |
| Lim3: Relies on human know-how and interpretation of the work products being reviewed, so it can be subjective (different people have different interpretation) | - |

**Create a diagram with this order:**

* **Statement Turn on the computer**
* **Start Google Email**
* **If Google email open then**
* **Send a letter**
* **Close Google Email**

Two test cases to test all the decisions!

****

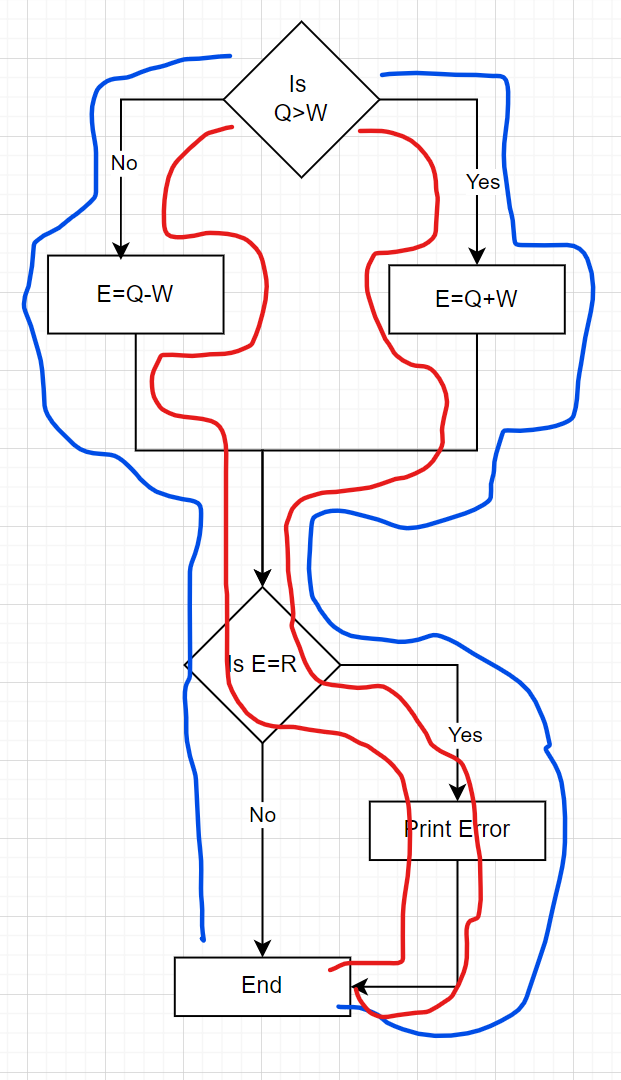
**Create a diagram with this order:**

* **If Q>W**
* **Then E=Q+W**
* **Else E=Q-W**
* **End**
* **Read R**
* **If E=R**
* **Then print “Error”**
* **End**

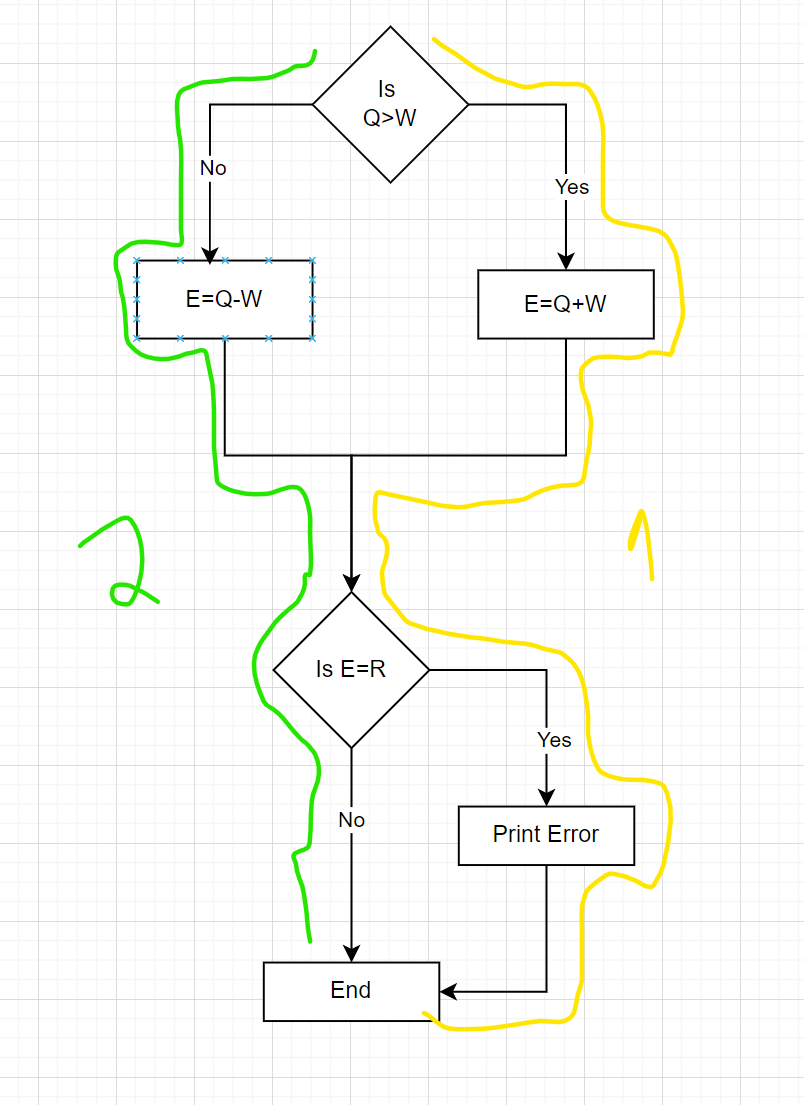
**How many test cases are needed to cover all the decisions in this code?**

2 for all the statements

2 for all the decisions



* With a total of test cases we can test all the statements and all the decisions.



**When the code has a single 'IF' condition and no loops (LOOP) or switches (CASE), any test we run will result in 50% decision coverage.**

**a. Correct. Any test case provides 100% coverage of statements, thus**

**covering 50% of solutions.**

**b. Correct. The result of any IF condition test will be either true or false.**

**c. Incorrect. A single test case can guarantee 25% coverage of the solutions**

**in this case.**

**d. Incorrect, because it is too general a statement. We cannot know if it is correct as it depends on the software being tested.**

Correct Answer is B: We will run 50% of the decision coverage because the result of any if condition test will be either true or false.

**There’s the following pseudocode:**

**Switch PC on -> Start MS Word -> IF MS Word starts THEN -> Write a poem -> Close MS Word.**

**How many test cases will it take to test its functionality?**

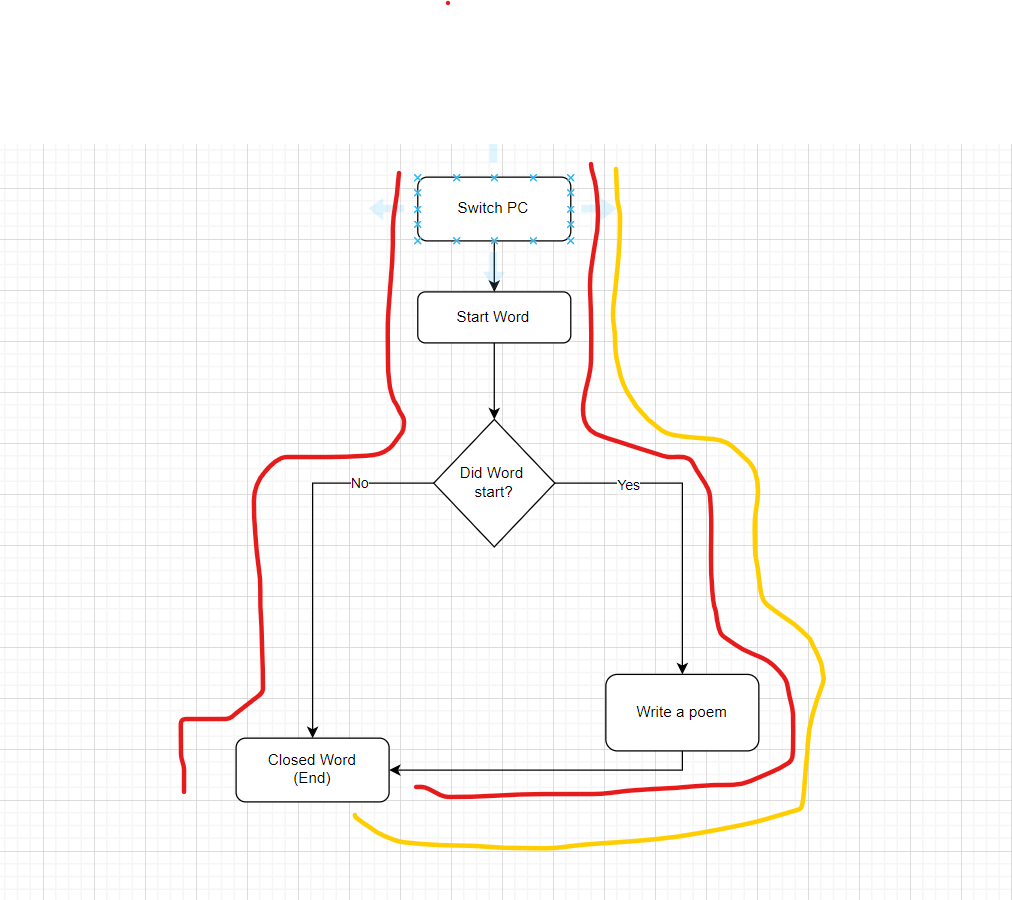
**a. 1 for operator coverage, 2 for decision coverage**

**b. 1 for operator coverage, 1 for decision coverage**

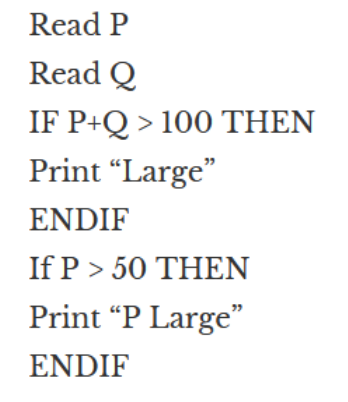
**c. 2 for operator coverage, 2 for decision coverage**

**d. 2 for operator coverage, 1 for decision coverage**

Correct answer is A: because we just new one test to have operator coverage (highlighted in yellow), but we need two tests to have decision coverage.



**How many tests are needed to check code statements:**



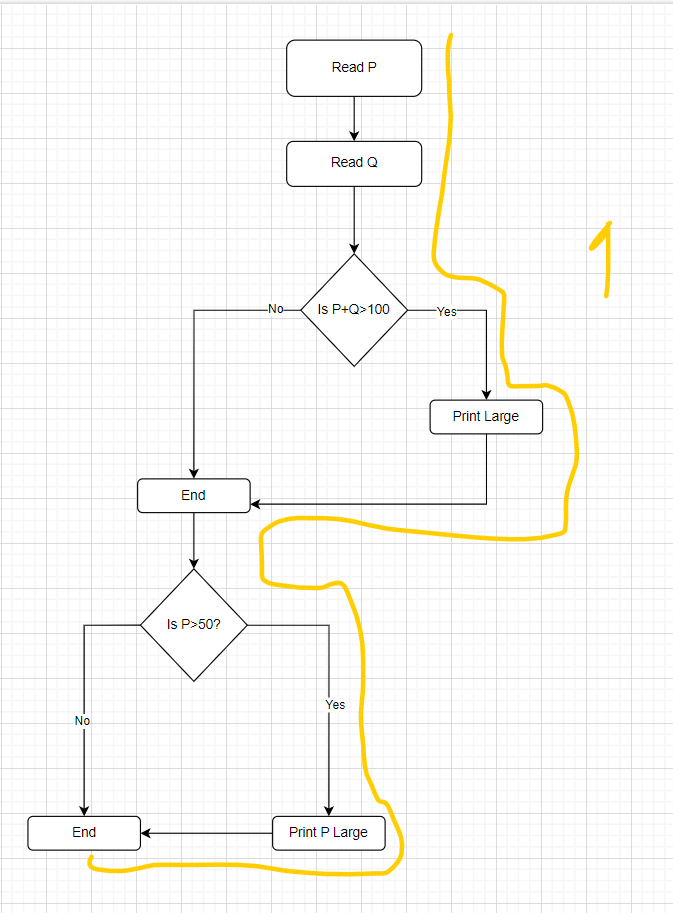
a. 2

b. 1

c. 3

d. 4

Correct answer is 1: One test to check all the code statements, like the image below



There’s the following algorithm:

Ask what kind of pet the user has.

* If the user answers that they have a cat, then ask what breed it is: "short-haired or
* long-haired?"
  + If the user answers "long-haired", then ask: "Would you like to get the contacts of the nearest groomer?"
  + If the user answers "yes", then say: "Give me the address of the nearest cat grooming salon."

else

* + Say: "Suggest a shop with fur care products"
  + end
  + else
  + Say "Suggest a pet shop"
  + end
* If the user has no cat
* Say "Come back when you decide to get a cat"
* end

Draw an algorithm diagram (in a tool of your choice, for example, in the built-in

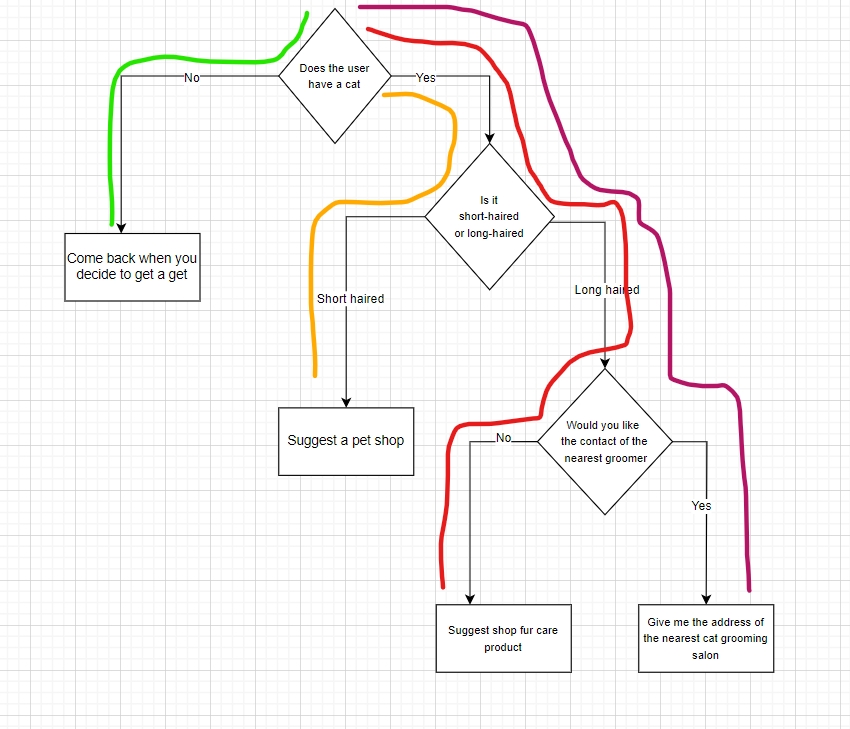
Google Docs editor, figjam, or other.)

What is the minimum set of test cases needed to make sure that all questions

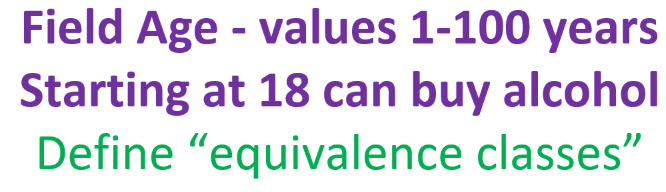
have been asked, all combinations have been passed, and all answers have been

obtained?

The minimum of test case to run is four, to make sure that we conver all decisions and all statements.



**Define “equivalence classes”:**

****

**Class 1:** Negative number;

**Class 2:** 0;

**Class 2:** Ages from 1 to 17;

**Class 3:** 18 to 100;

**Class 4:** 101 and above

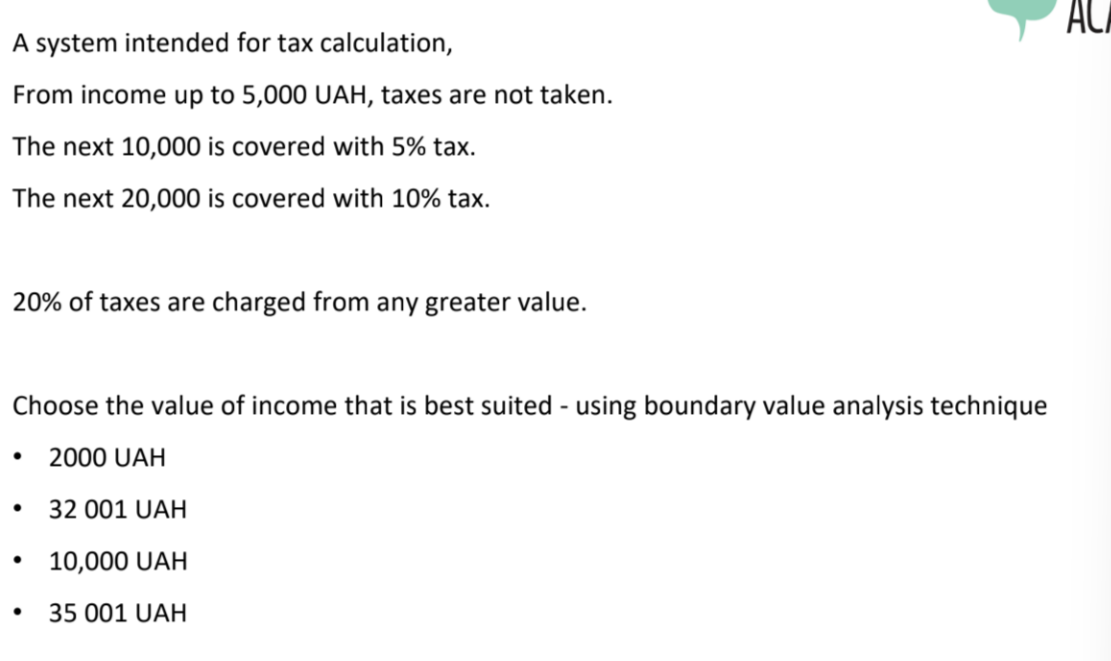
**Define the “Boundary Vales”**



Boundary Value:

* 199
* 200
* 201

**Define the “Boundary Vales”**



Classes:

Income 1: 0,000$ to 5,000$ - No Taxes

Income 2: 5,001$ - 15,000$ - Taxes of 5%

Income 3: 15,001$ - 35,000$ - Taxes to 10%

Income 4: 35,001$

* Boundary Value is $35,001

**1. The speed control system has the following characteristics:**

* at a speed of 50 km/h or less — the system does not respond
* at a speed of more than 50 but less than 55 km/h — the system issues a warning
* at a speed of more than 55, but less than 60 km/h — the system will issue a fine
* at a speed of more than 60 km/h — the driver will receive a fine and a penalty point in the driver's license

**Class 1: >=50km/h**

* Boundary Level: 49,50, 51

**Class 2: 50.1km/h to 55km/h**

* Boundary Level. 54,55,56

**Class 3: 55.1km/h to 59km/h**

* Boundary Level: 58,59,60

**Class 5: 60 or more km/h**

* Boundary Level: 59,60,61

**The speed in the system is measured in integer values. Which of the test data sets can be**

**used to test all the boundary values of the equivalence classes?**

**A. 0, 49, 50, 54, 59, 60**

**B. 50, 55, 60**

**C. 49, 50, 54, 55, 60, 62**

**D. 50, 51, 55, 56, 60, 61**

For A: We don’t have a boundary level for Class 5 with higher than 60km/h

For B: Not all classes are here

For C: 62 is not a boundary level

**Correct Answer is D**

**A fitness app counts the number of steps and sends messages to encourage the users to**

**move. Depending on the number of steps, the feedback will be as follows:**

**- up to 1000 inclusive – “Couch potato”**

**- from 1000 to 2000 inclusive – “Bit lazy, huh”**

**- from 2000 to 4000 inclusive – “Keep moving!”**

**- from 4000 to 6000 inclusive – “Nice!”**

**- over 6000 – “Amazing job!”**

**Which test data set will provide the best coverage of equivalence classes?**

**A. 0, 1000, 2000, 3000, 4000**

**B. 1000, 2001, 4000, 4001, 6000**

**C. 123, 2345, 3456, 4567, 5678**

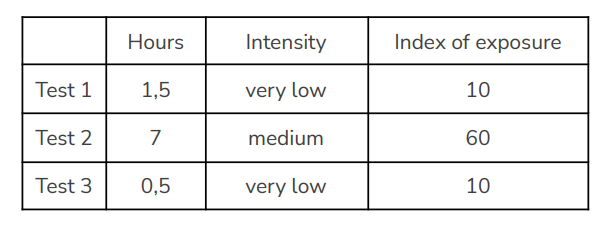
**D. 666, 999, 2222, 5555, 6666**

**Correct Answer is D: In A,B,C we are just testing three classes but in D we are testing four classes**

**2. A device that measures the time and intensity of sunlight received by a plant counts a**

**combination of parameters: time in the sun (less than 3 hours, 3 to 6 hours, and more than 6**

**hours) and light intensity (very low, low, medium, high). There is the following set of tests:**

****

**What is the minimum number of additional test cases required to ensure that all valid equivalence classes are covered?**

**A. 1**

**B. 2**

**C. 3**

**D. 4**

**Correct Answer is B**

Missing right now: A test for more than 6 hours and a test to measure intensity for low and high. With this in mind we need to have two additional tests one to test more than 6 hours and intensity high and a second one just to test the intensity low.

**The video playback application has requirements. The application will work on devices**

**with this resolution:**

* **640x480**
* **1280x720**
* **1600x1200**
* **1920x1080**

**What test case is the result of applying the equivalence partitioning technique? Support your answer.**

**A. Check that the application plays video on a 1920x1080 display (1 test case)**

**B. Check that the application plays video on 640x480 and 1920x1080 displays (2 test**

**cases)**

**C. Check that the application plays video on displays of all sizes specified in the**

**requirements (4 test cases)**

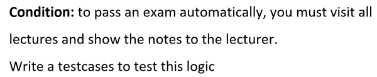
**D. Check that the application plays video on any display size specified in the**

**requirements (1 test case)**

**Correct Answer is C**

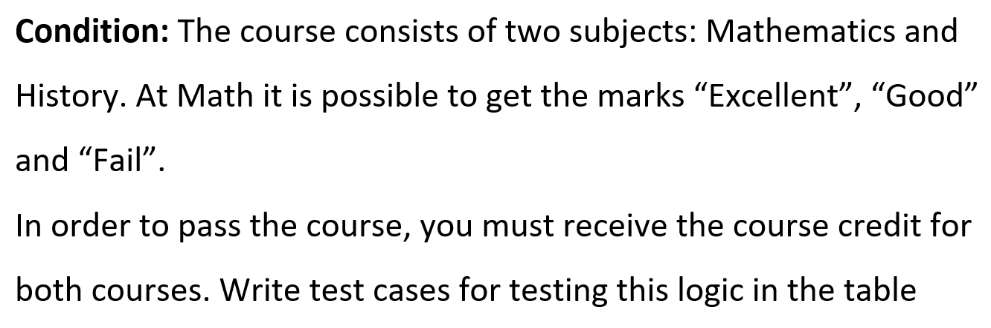
There are four classes for each resolution, that is the basis of equivalence partitioning, so we need to run four test cases that will represent each class.

**Build a decision table based on this:**

****

|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** |
| --- | --- | --- | --- | --- |
| Attendance | 1 | 1 | 0 | 0 |
| Notes | 0 | 1 | 1 | 0 |
| Exam | 0 | 1 | 0 | 0 |

**Build a decision table based on this:**

****

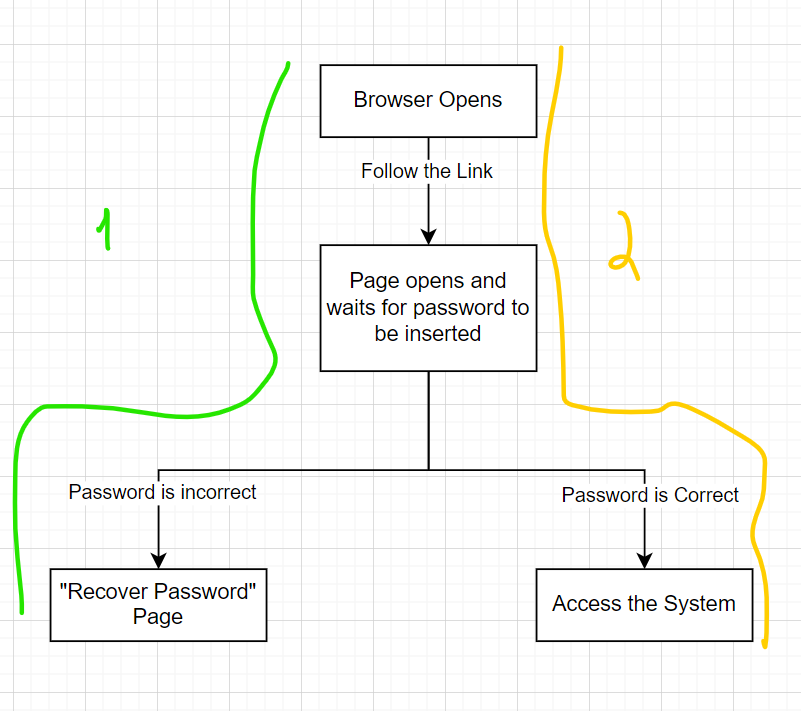
|  | Test Case 1 | Test Case 2 | Test Case 3 | Test Case 4 | Test Case 5 | Test Case 6 |
| --- | --- | --- | --- | --- | --- | --- |
| Math Excellent | 1 | 0 | 0 | 1 | 0 | 0 |
| Math Good | 0 | 1 | 0 | 0 | 1 | 0 |
| Math Failed | 0 | 0 | 1 | 0 | 0 | 1 |
| History Pass | 1 | 1 | 1 | 0 | 0 | 0 |
| History Fail | 0 | 0 | 0 | 1 | 1 | 1 |
| Passed the Course | 1 | 1 | 0 | 0 | 0 | 0 |

**Condition: To enter the system, you must follow the link and enter a**

**password. If you enter an incorrect password, you redirects to the**

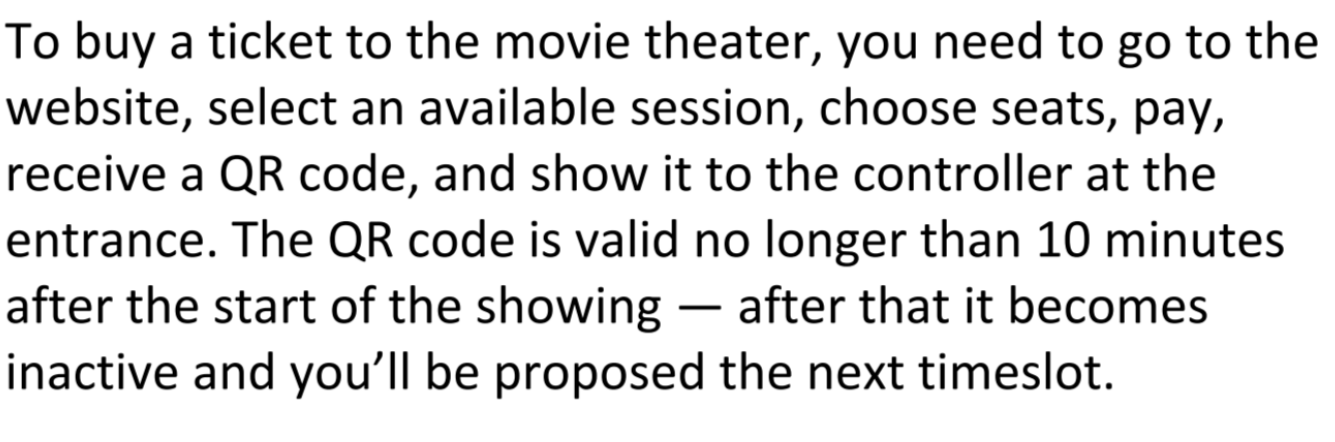
**"recover password" page.**

**Compose test cases to test this functionality**

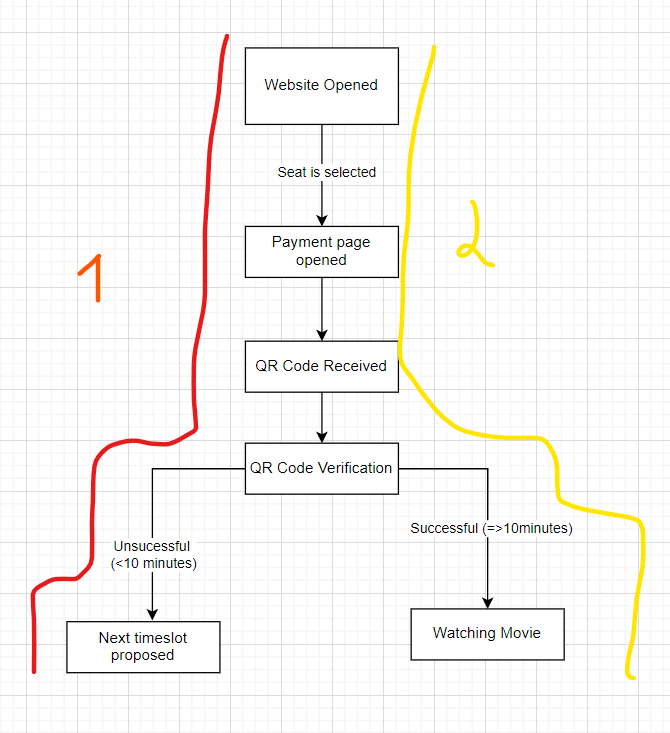
****

We need to develop two test cases for this functionality.

**Condition:**

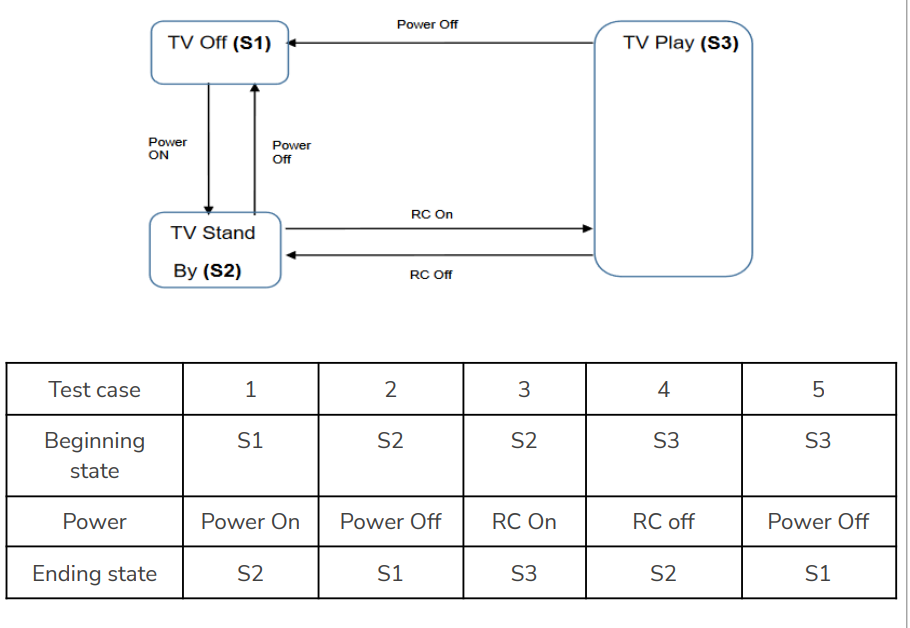
****

**How many test cases are necessary?**

****

Two test cases are necessary, as shown in the screenshot above.

**Which statement about the state transition diagram and the table with test cases is correct?**

****

**A. These test cases cover valid and invalid transitions in the chart.**

**B. These test cases show all possible valid transitions on the chart.**

**C. These test cases cover some valid transitions on the chart.**

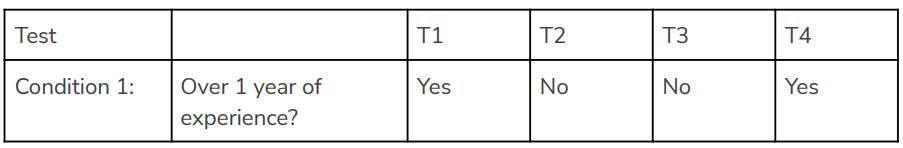
**D. These test cases cover pairs of transitions in the diagram.**

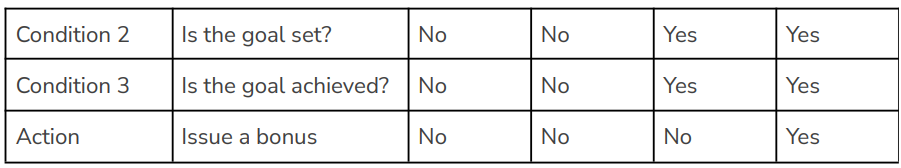
Correct answer in my opinion is answer B.

**Employees of the company are issued bonuses if they have worked for more than a**

**year and have fulfilled the goals set in advance.**

**These conditions can be presented in a decision table:**

****

****

**Which scenario that is likely in real life is omitted in the table?**

**A. Condition 1 = YES, Condition 2 = NO, Condition 3 = YES, Action = NO**

**B. Condition 1 = YES, Condition 2 = YES, Condition 3 = NO, Action = YES**

**C. Condition 1 = NO, Condition 2 = NO, Condition 3 = YES, Action = NO**

**D. Condition 1 = NO, Condition 2 = YES, Condition 3 = NO, Action = NO**

Correct answer in my opinion is answer D.

The first one is impossible, because a goal can’t be achieved if it wasn't set, the second one is impossible because the bonus can be issued if we didn’t achieve the goal, and the third one is the same, the goal can’t be achieved if it wasn't set.

**Make a state and transition diagram for testing the following video game:**

**After starting the game, you find yourself in the Castle of Mysteries, in a room with two**

**corridors — to the right and to the left.**

**If you go to the right, you will get to the dragon. The dragon gives you a riddle. If the**

**answer is correct — you leave the castle and win. If not, the dragon gives another riddle.**

**If you answer incorrectly for the second time, the dragon will eat you and you lose.**

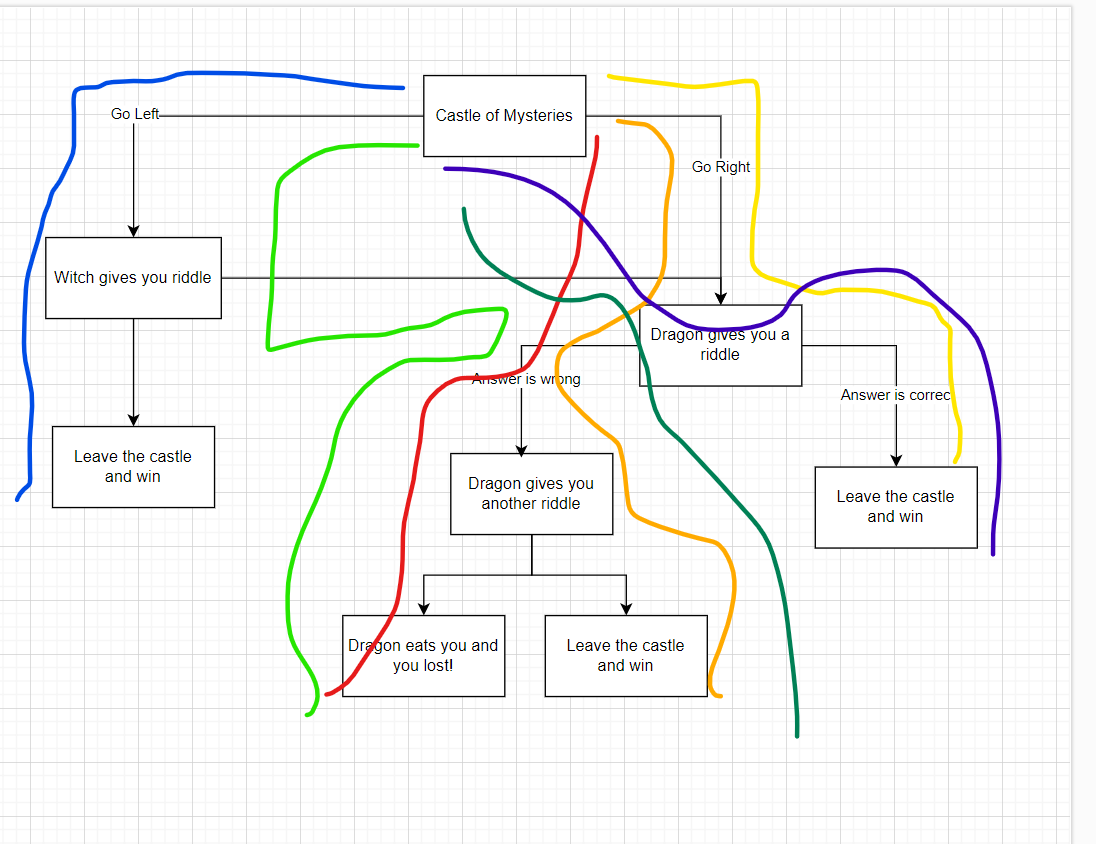
**If you go to the left, you will get to the witch. The witch gives a riddle. If the answer is**

**correct, you leave the castle and win. If the answer is wrong, the witch takes you to the**

**dragon.**

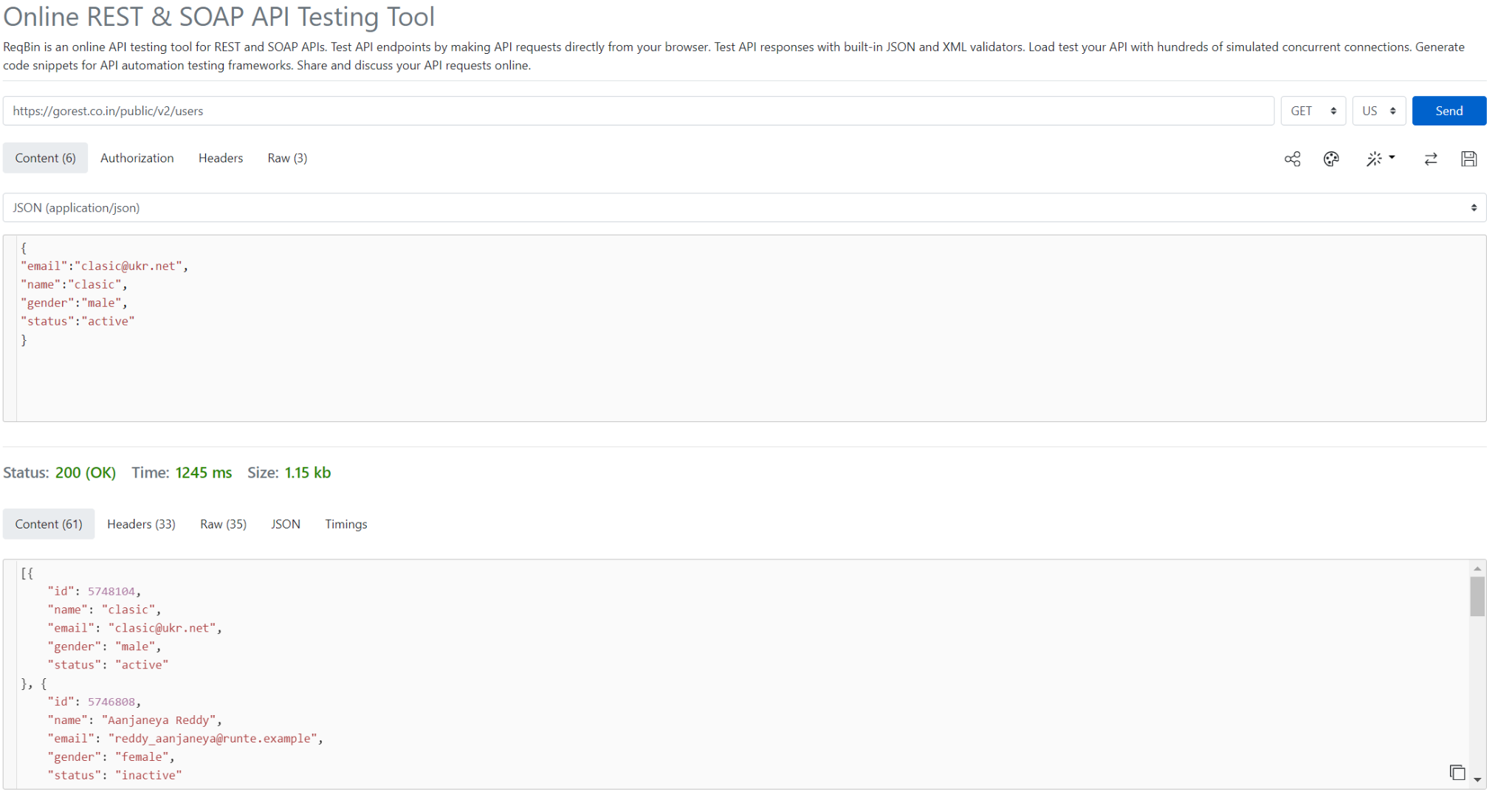
**How many test cases, according to the diagram, will be enough to test this game?**

We have in total seven test cases that we need to for test the whole game represented in the state transition diagram.

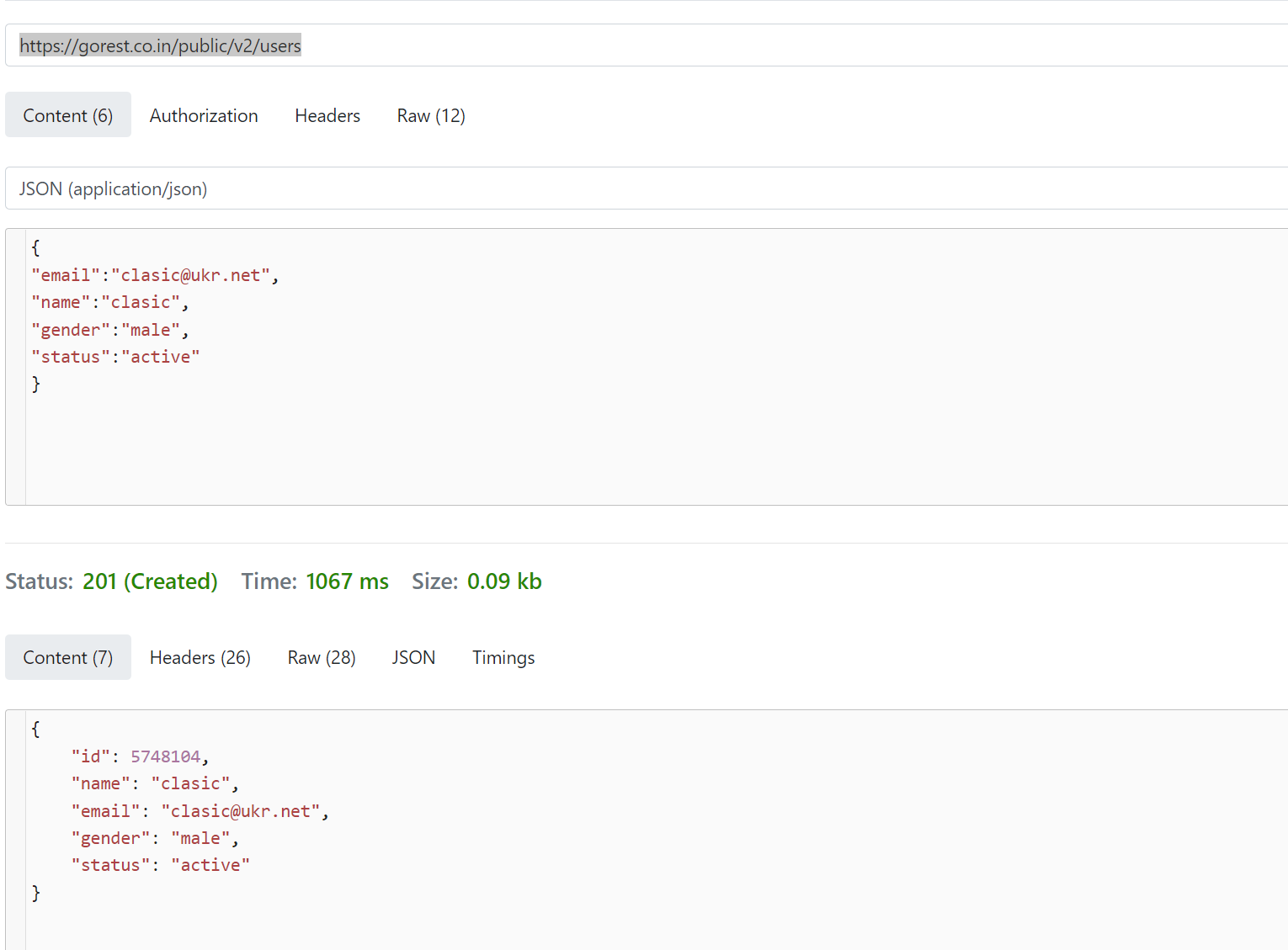


**Do the following requests for this API:**

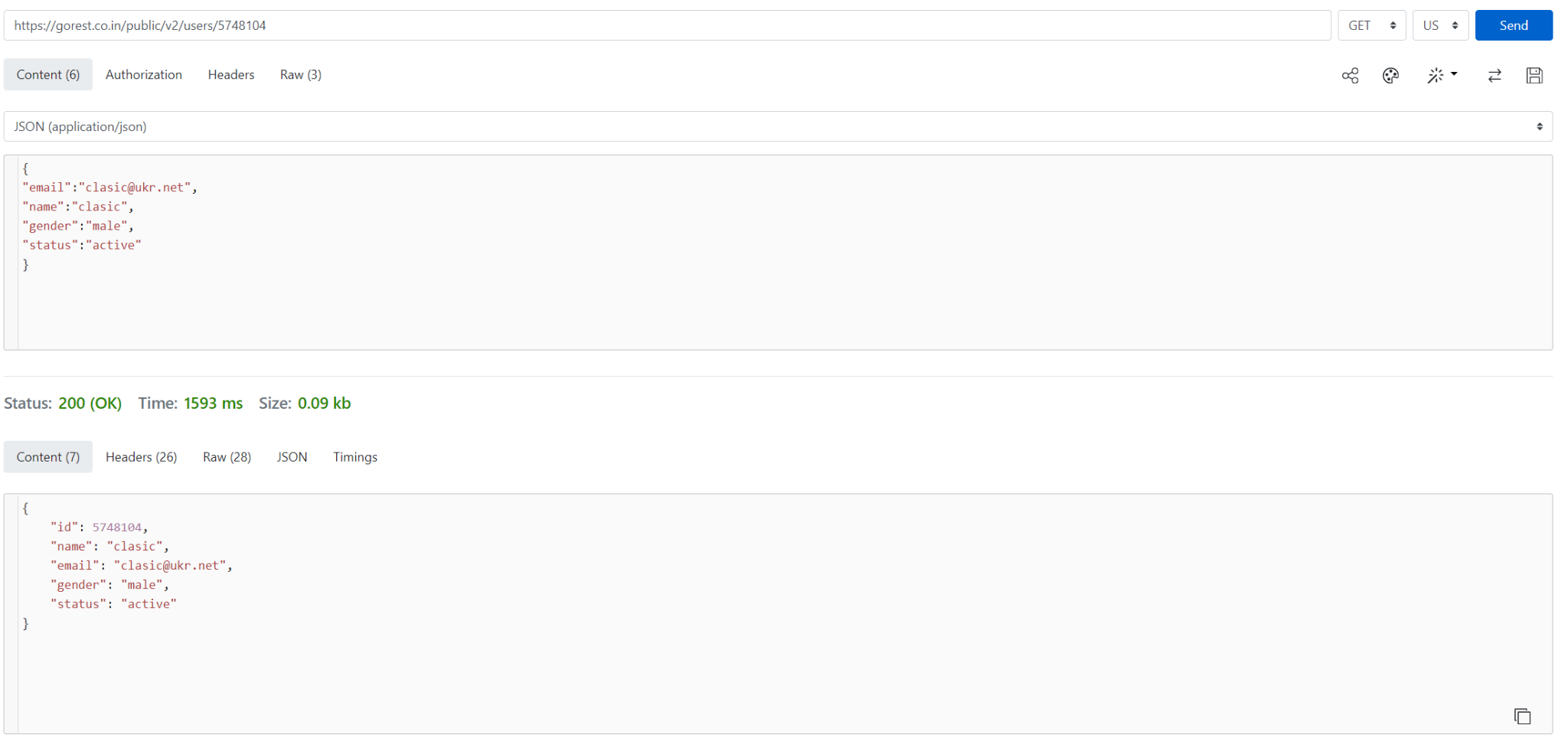
* + **https://reqbin.com/ - for requests**
  + **https://gorest.co.in/ - for users creation**
* **Get All Users**

****

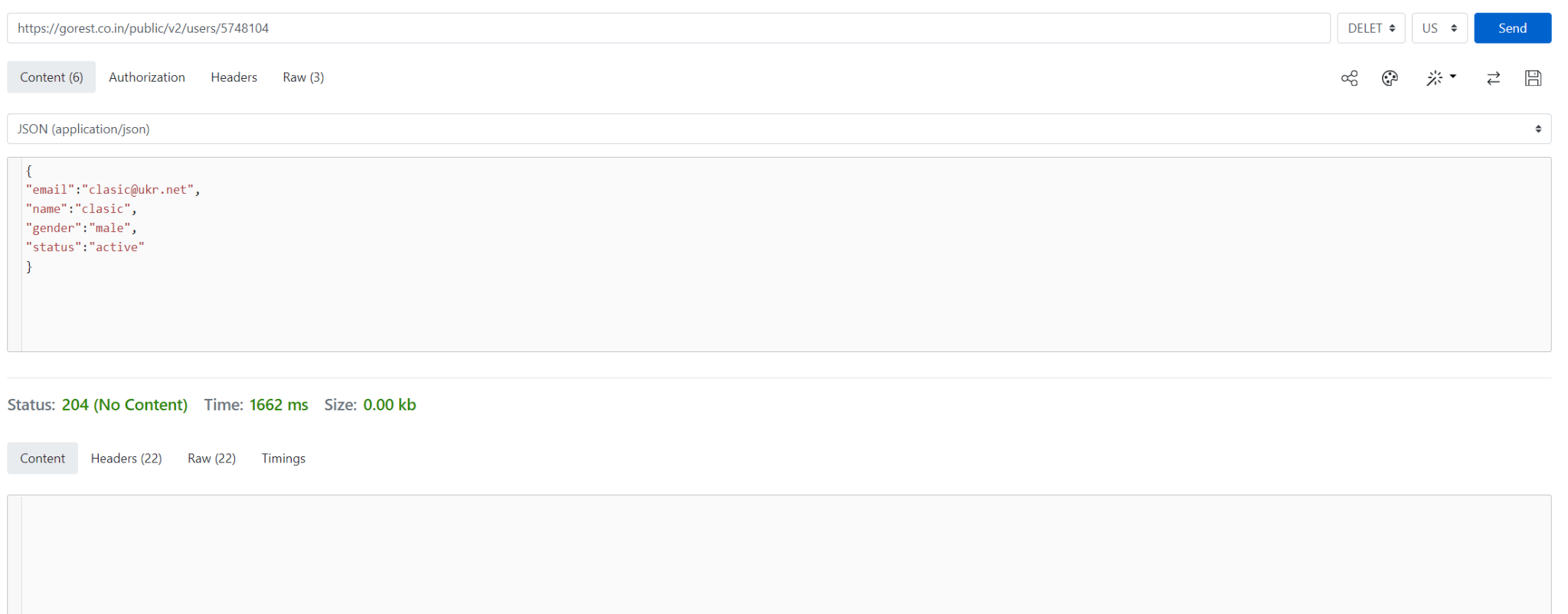
* **Post A User**

****

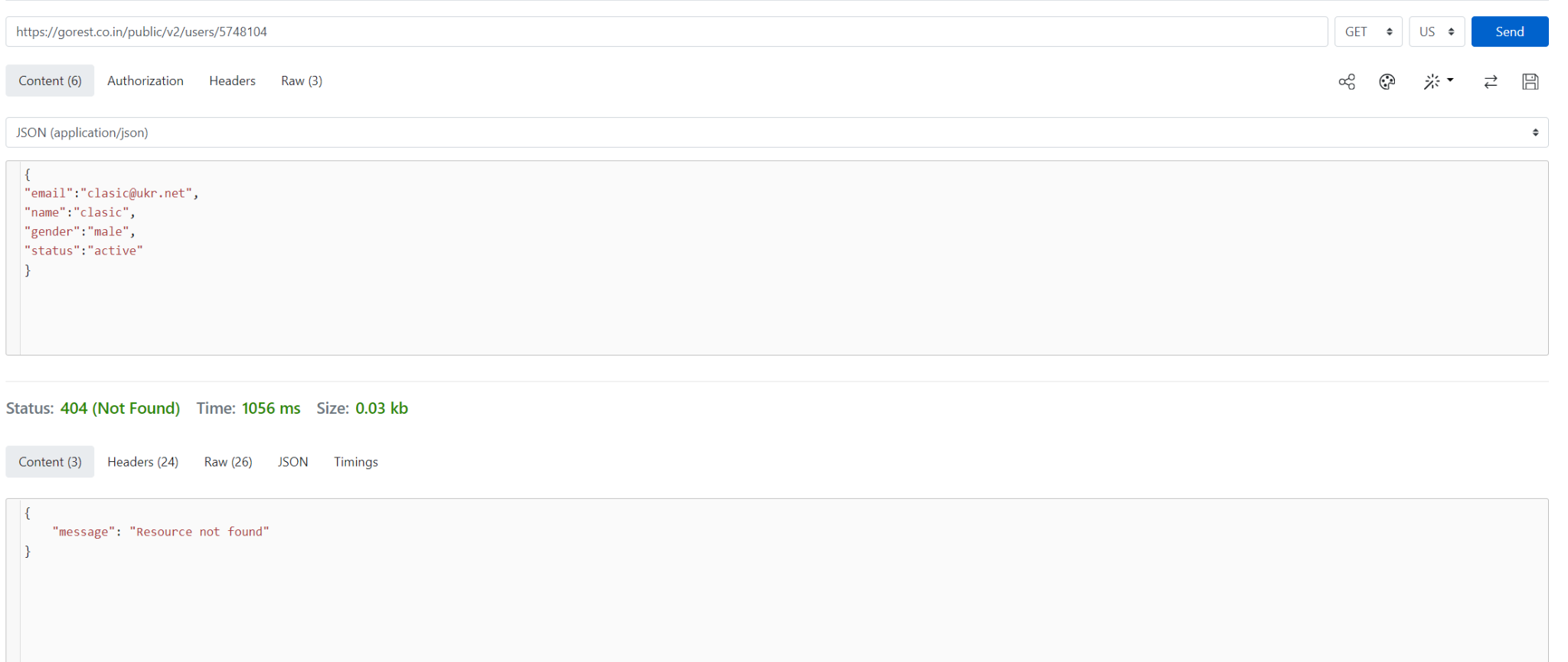
* **Get Information from That User**

****

* **Delete That User**

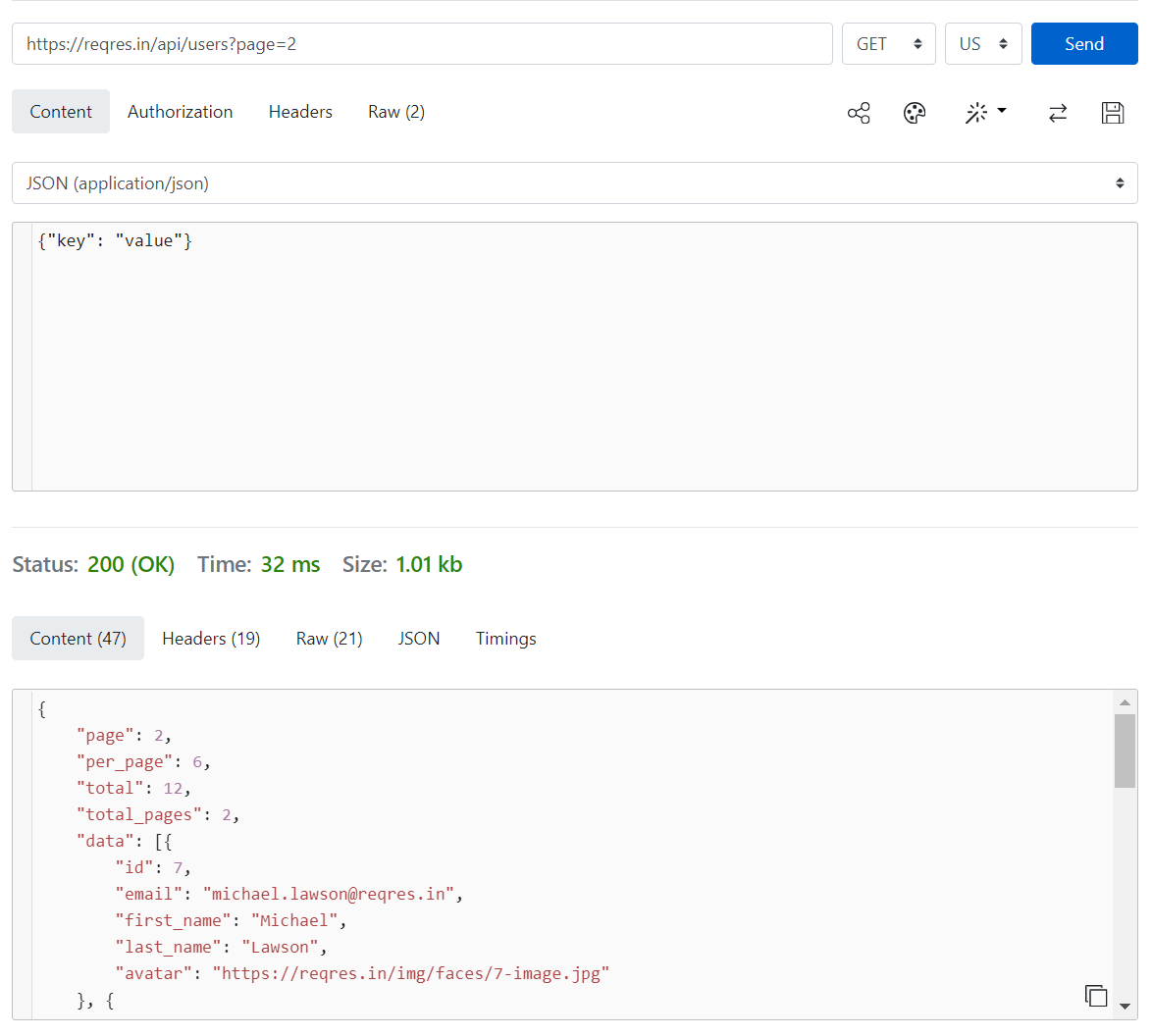
****

* **Get Information from the user deleted**

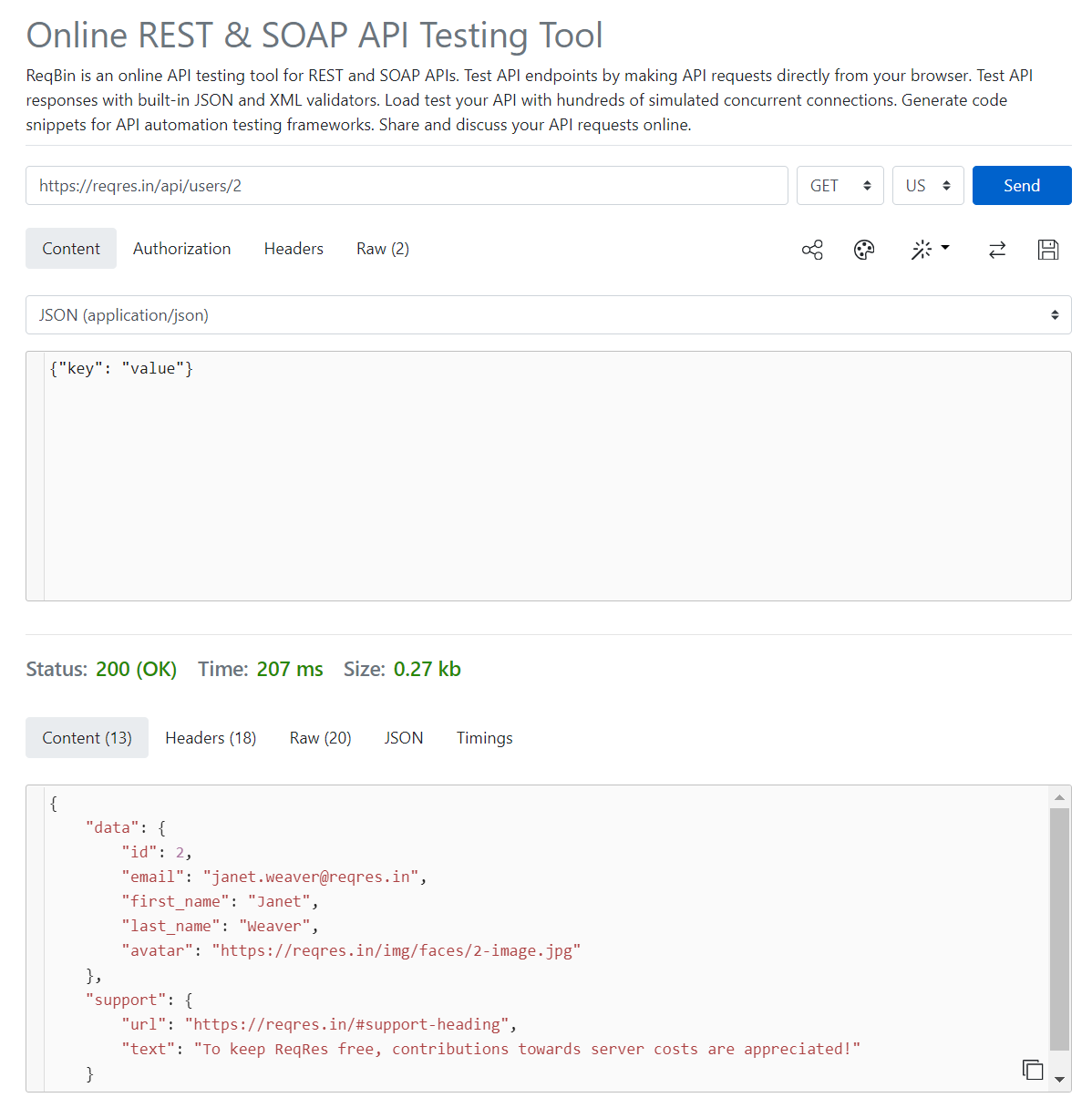
****

**With https://reqbin.com/, test the following API Reqres methods:**

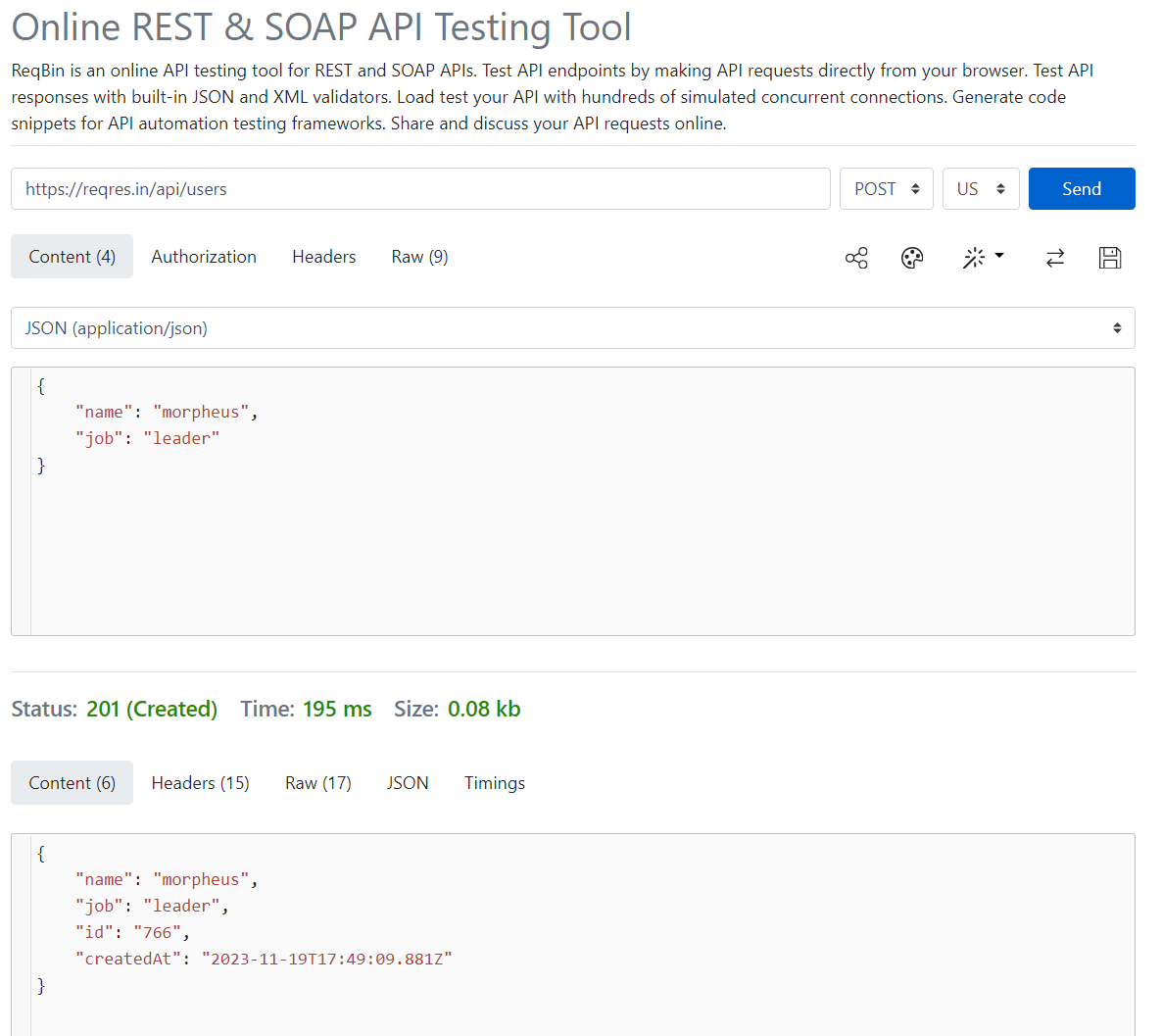
* **LIST USERS**

****

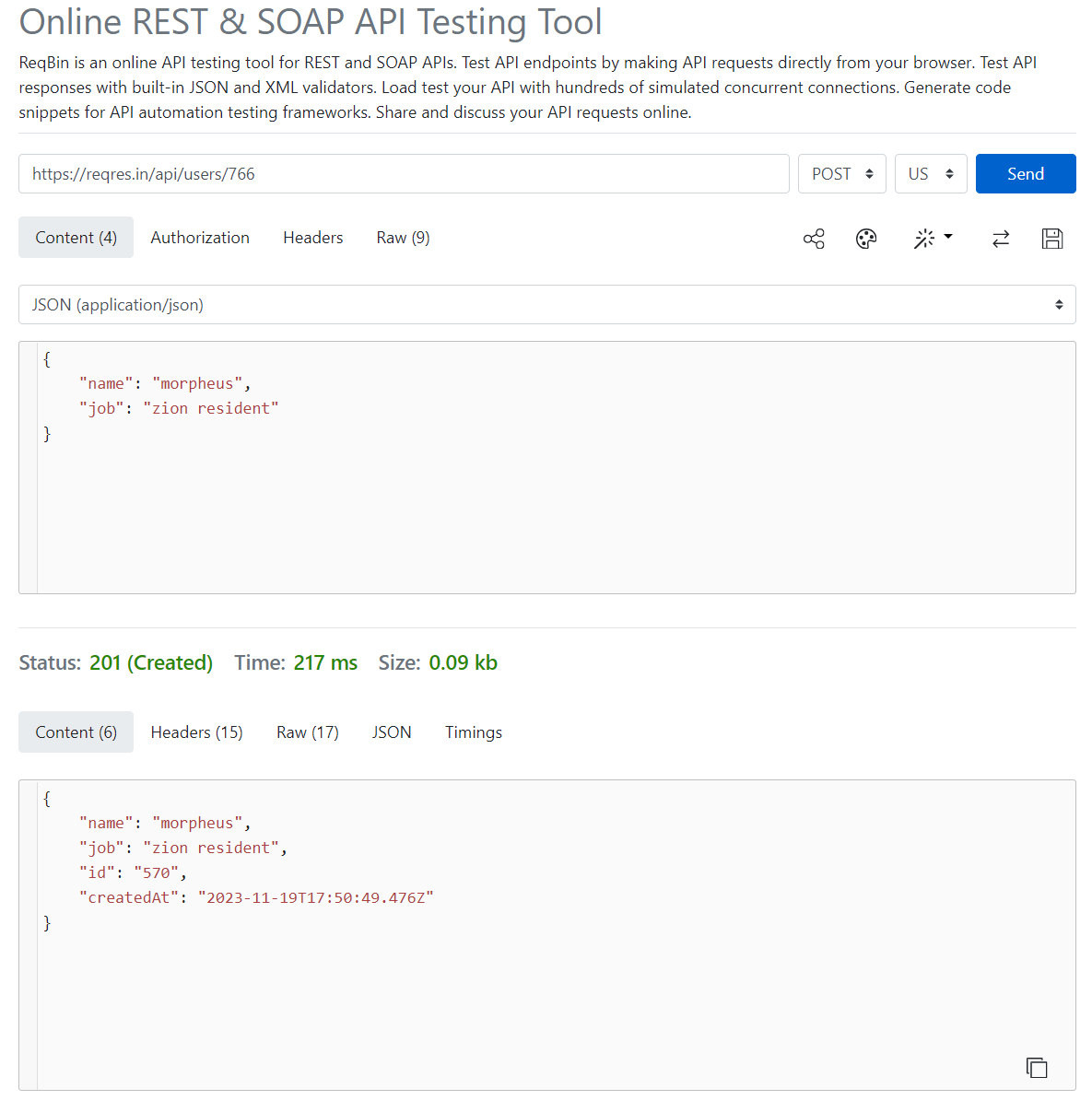
* **SINGLE USER**

****

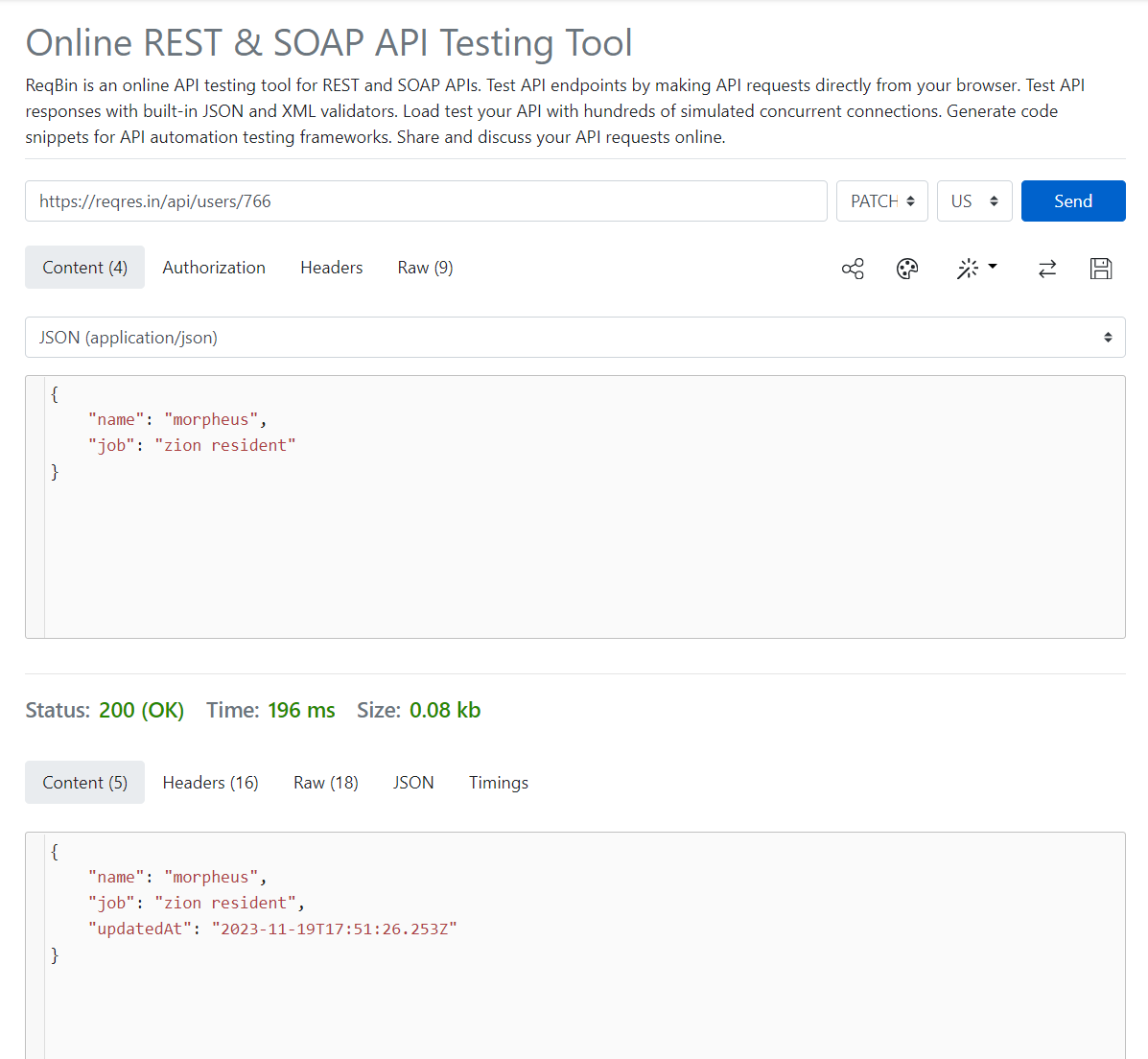
* **CREATE**

****

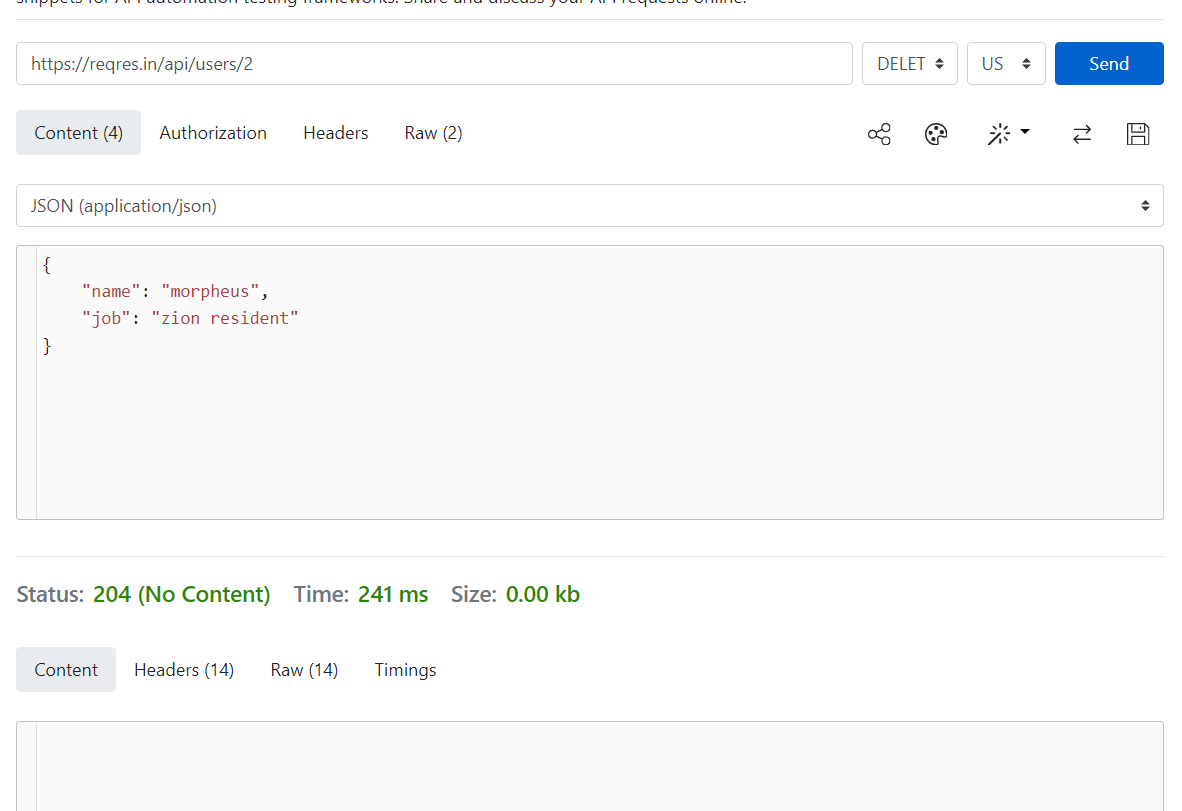
* **UPDATE (PUT)**

****

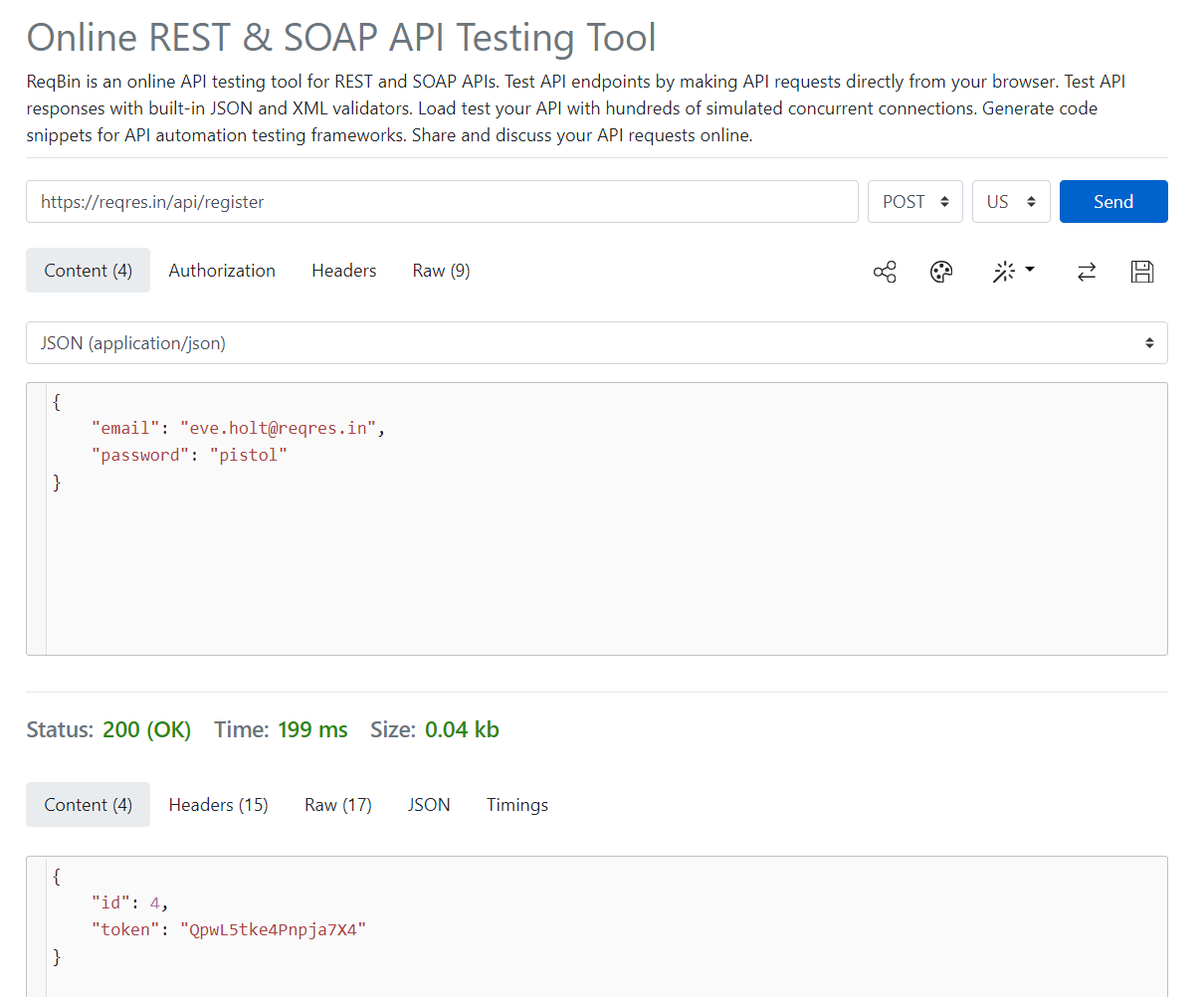
* **UPDATE (PATCH)**

****

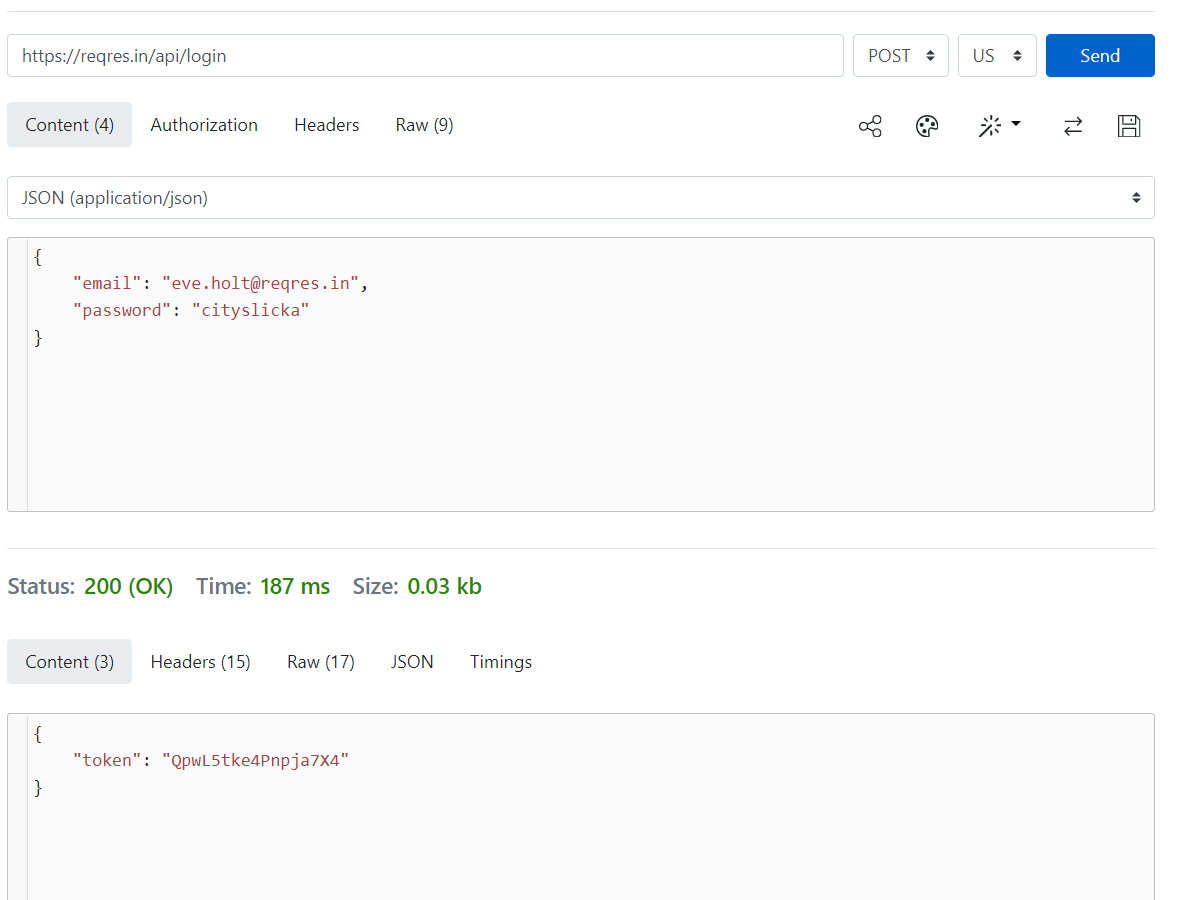
* **DELETE**

****

* **REGISTER - SUCCESSFUL**

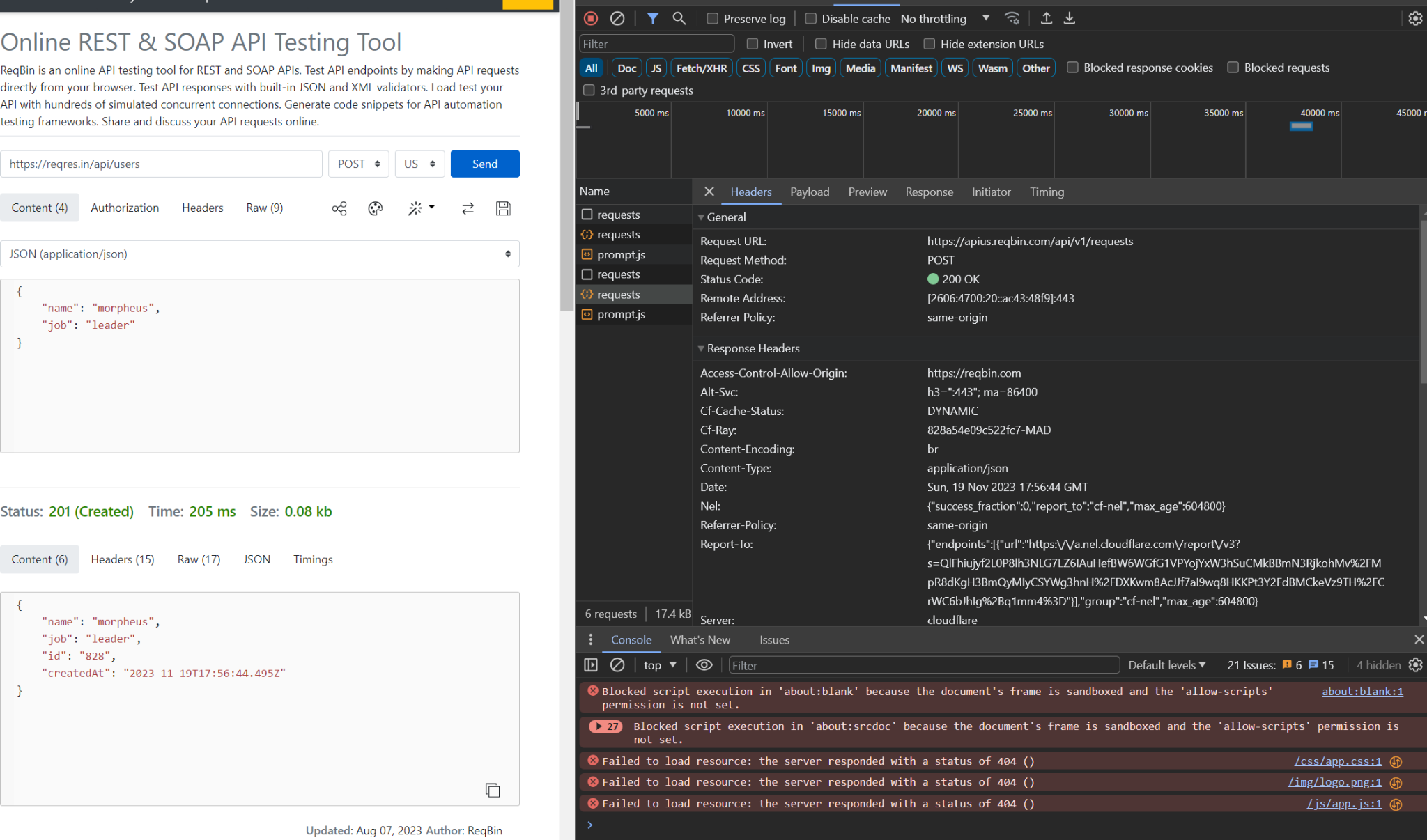
****

* **LOGIN - SUCCESSFUL**

****

**After sending a request, compare its details entered in https://reqbin.com/ and displayed in Developer Tools.**

I used a Post request, and you need to go to Network, headers and then on the request check for the method. Sometimes, it doesn’t match the request I’m making.

****

* **https://gorest.co.in/public/v2/users/ - POST**

**{**

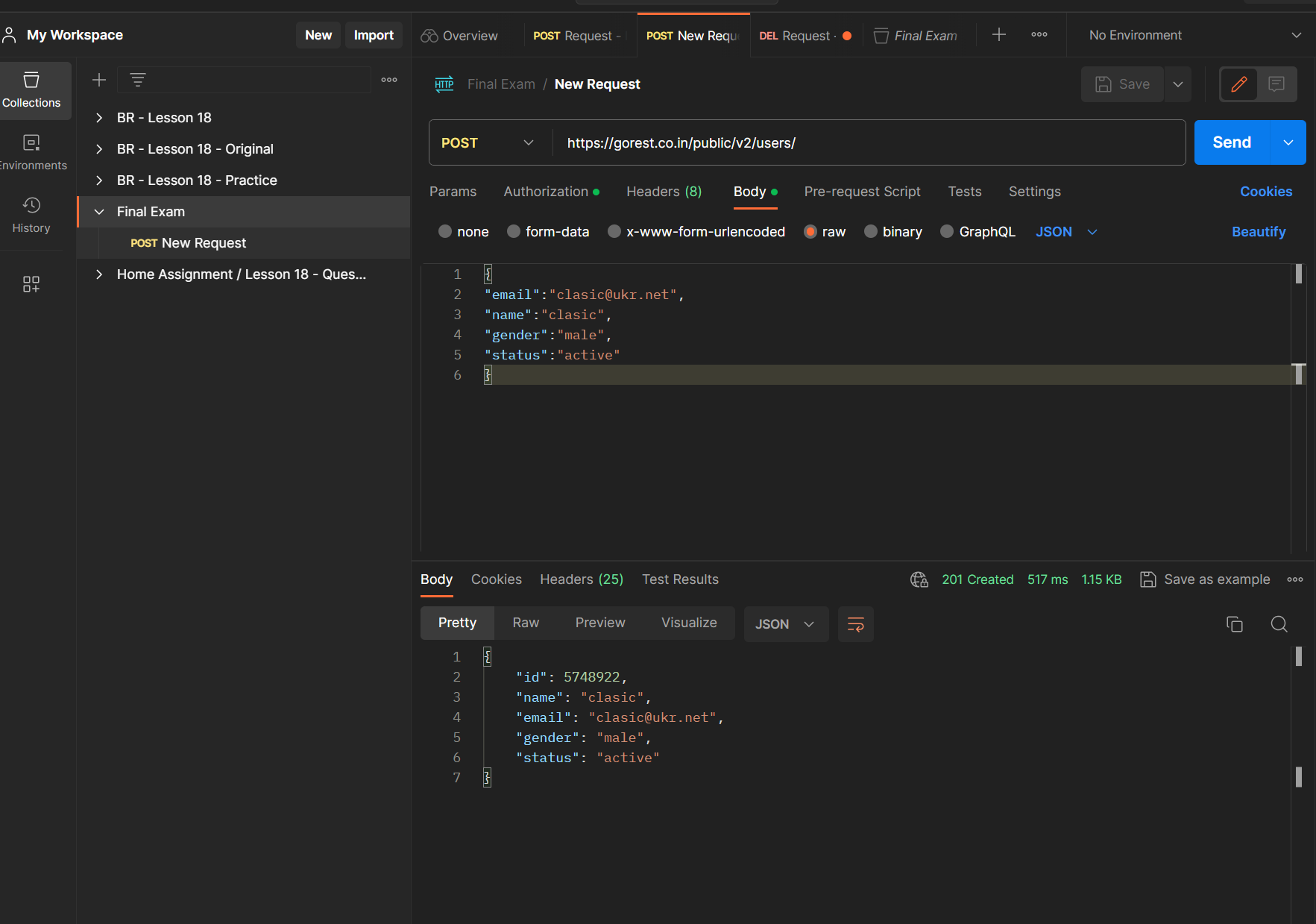
**"email":"clasic@ukr.net",**

**"name":"clasic",**

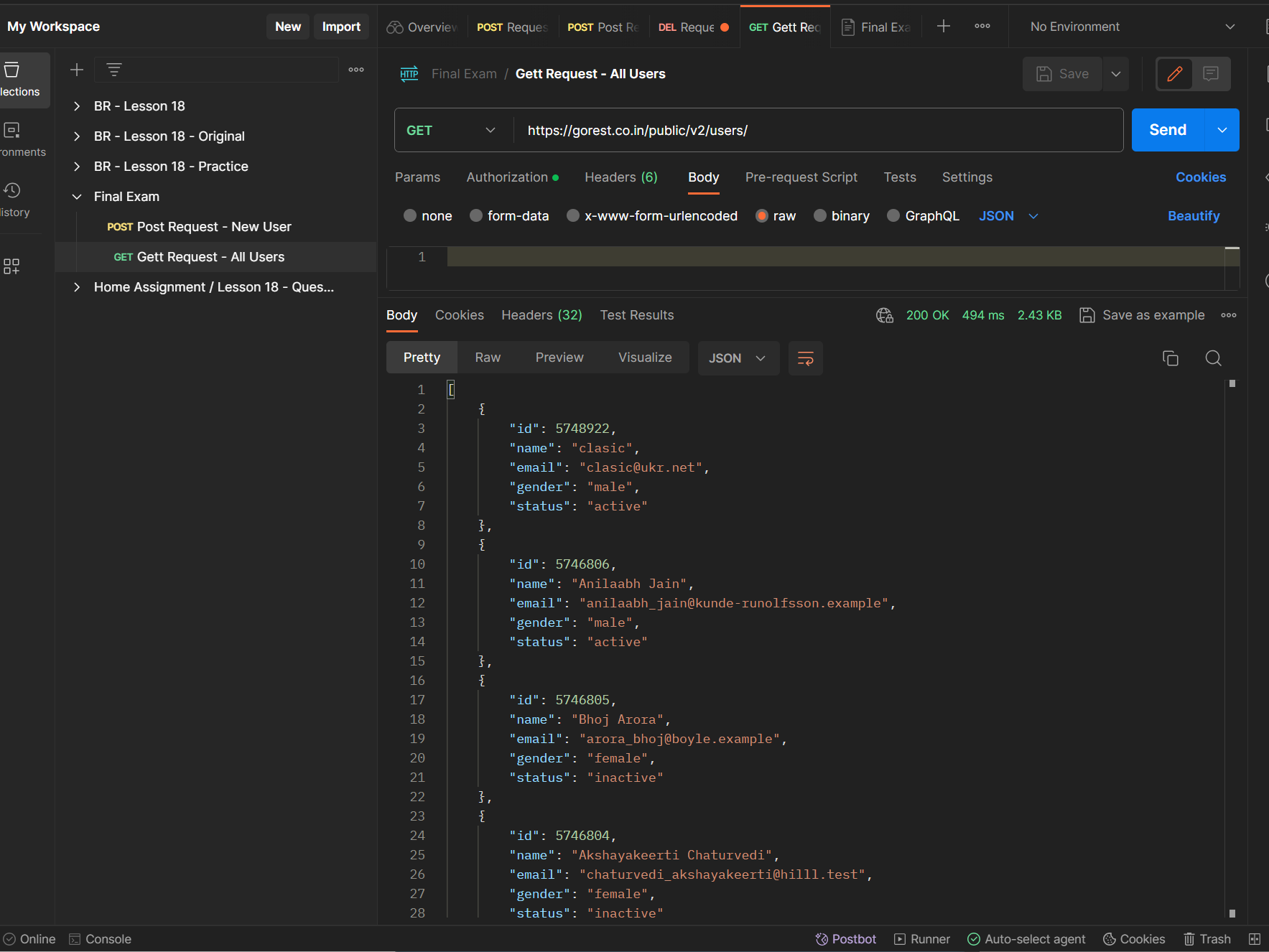
**"gender":"male",**

**"status":"active"**

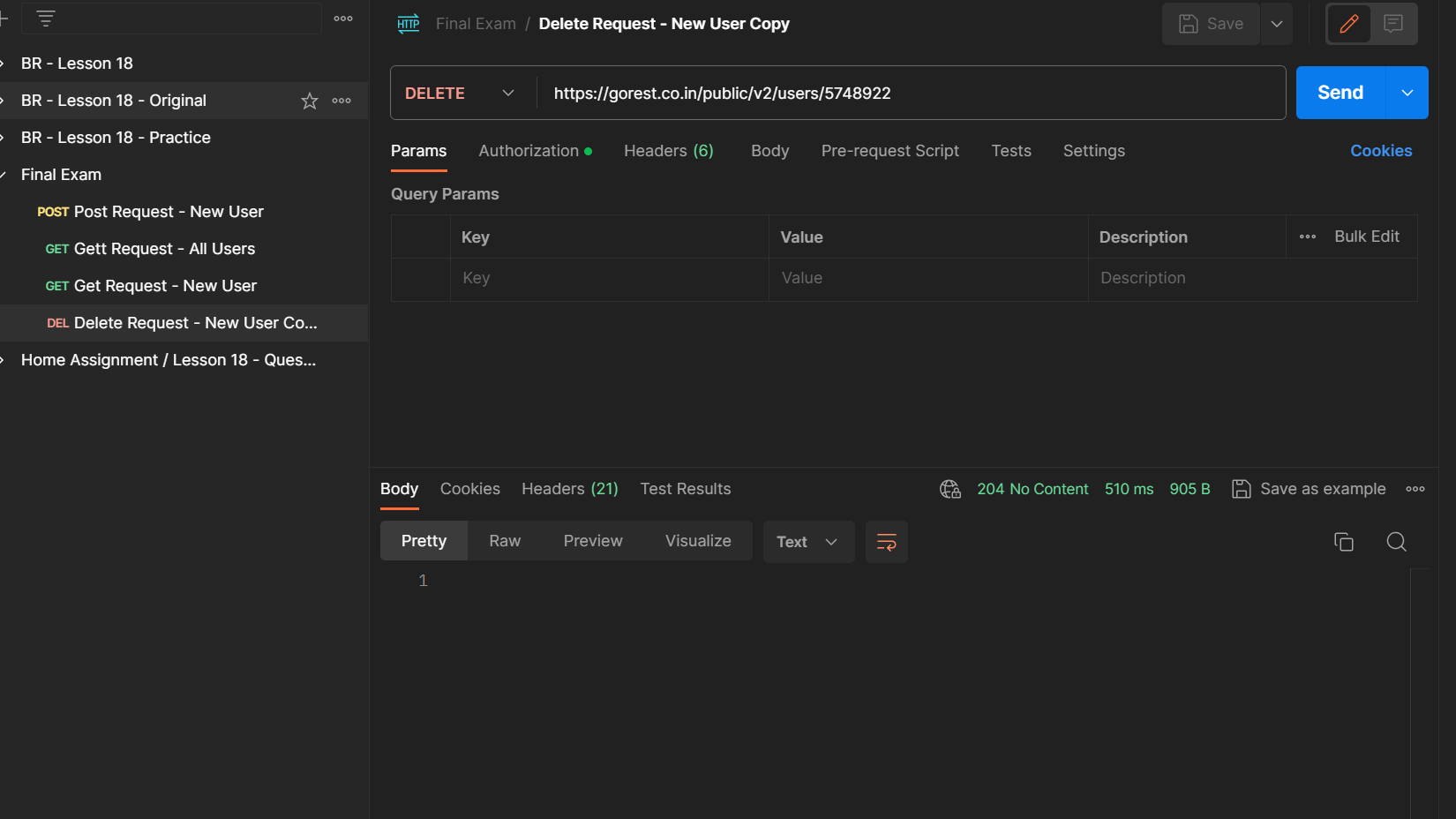
**}**

****

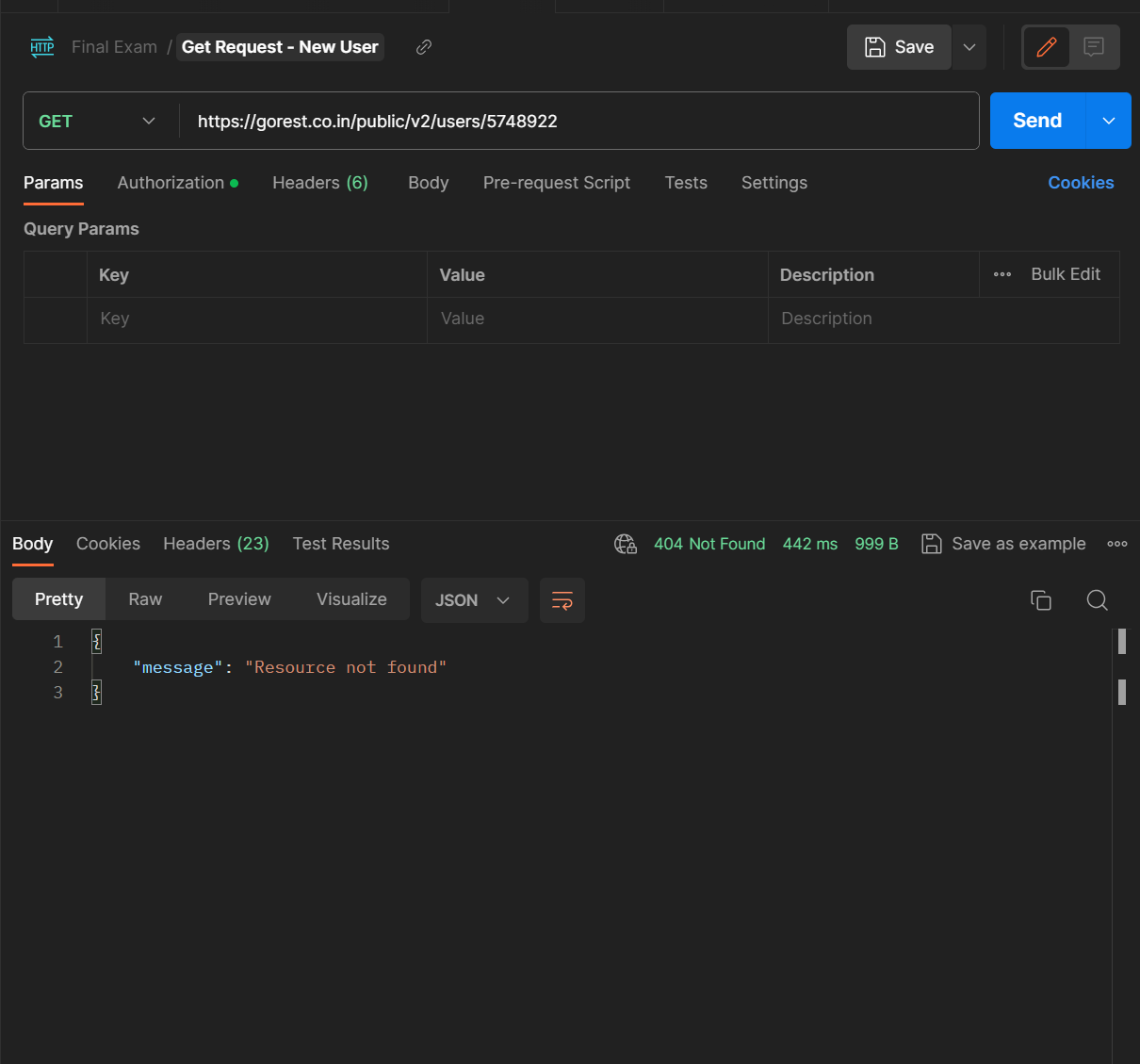
**https://gorest.co.in/public/v2/users/ - GET - all information**

****

**https://gorest.co.in/public/v2/users/5748922 - DELETE – users removed**

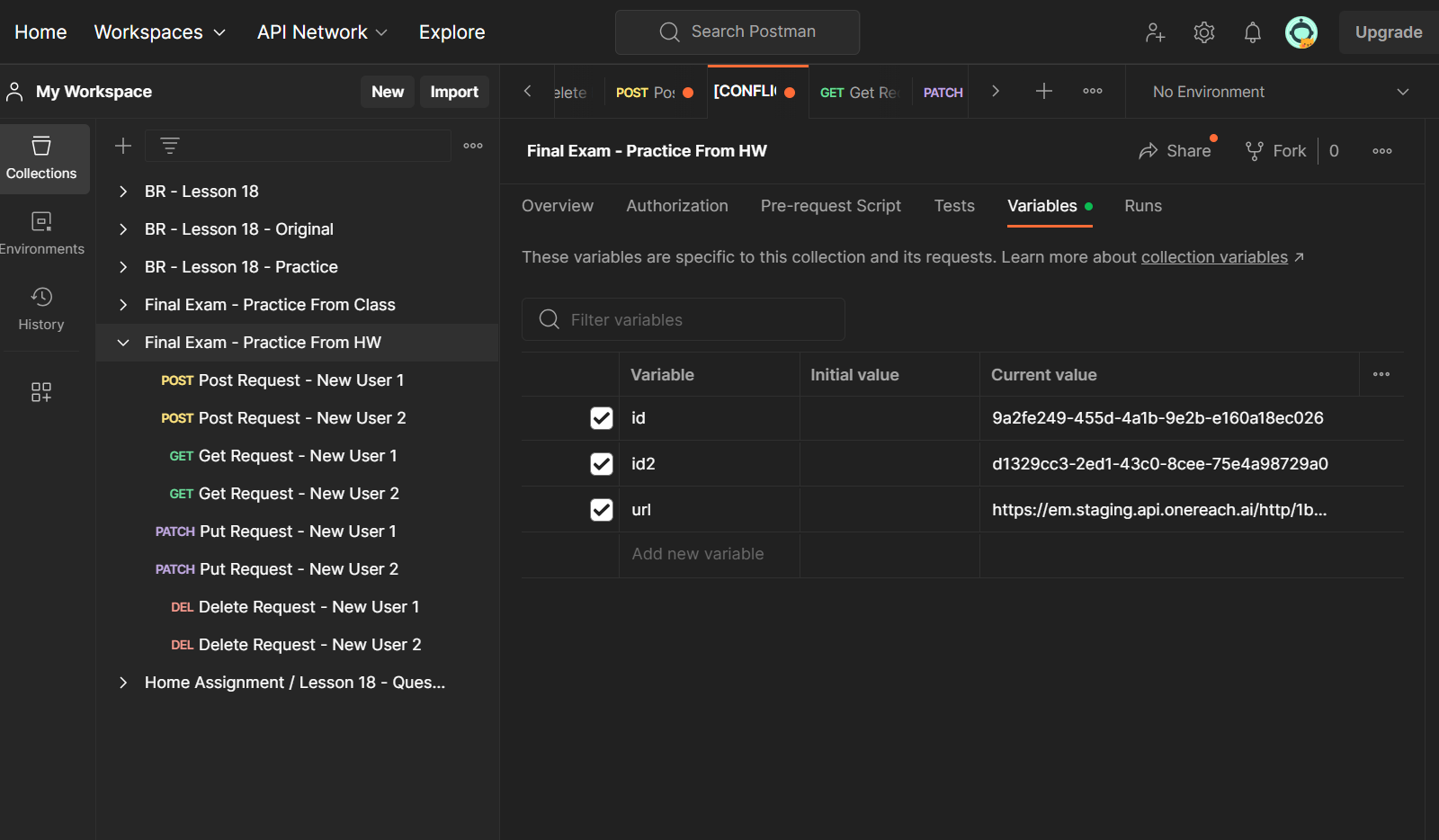
****

**https://gorest.co.in/public/v2/users/5748922 - GET - specific user user**

****

**Using API for practice.** [**Lesson 17.docx**](https://docs.google.com/document/d/1c6dOKbevG90jf2mwT-D90GCvEbV_NqS0/edit) **create POST, GET, PATCH, DELETE requests with:**

* **variables in the body of the request;**
* **variables in the request parameters;**
* **at least 2 tests for each request.**

****

**Create a repository on GitHub.**

**Add folders with the names of the previous lessons. In each folder, place the files with the completed assignments for the corresponding lesson.**

**Create a new repository on GitHub:**

* **clone it to a local repository;**
* **create an index.html file in the local repository;**
* **write a simple structure with key HTML tags in index.html;**
* **send the changes from the local repository to the remote one;**
* **attach a link to the remote repository and screenshots of the performed commands in LMS.**
* **Using the Bash shell’s CLI, perform the following actions:**
* **in your home directory, go to the Downloads directory**
* **create a GitLesson directory in it**
* **create a file called file1.txt in the Downloads directory**
* **write in file1.txt the text "Hello, world!" and save the changes**
* **display the contents of file1.txt on the screen**
* **copy file1.txt from the Downloads directory to the GitLesson directory and display the contents of the GitLesson directory**
* **add screenshots of the result of each command to LMS.**