Software Unit and System Test Documentation

Videogram

1.0.0

April 17, 2020

Computer Science Department, Biola University

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Primary Author(s) | Desc. Of Version | Date Completed |
| 1.0 | Derek James | Initial Version | 4/17/2020 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**1 INTRODUCTION 4**

1.1 Document Conventions 4

1.2 System Overview 4

1.3 Test Approach 4

**2 TEST PLAN 4**

2.1 Features to be Tested 4

2.2 Features not to be Tested 4

2.3 Testing Tools and Environment 5

**3 TEST CASES 5**

**3.n Case-n 5**

3.n.1 Purpose 5

3.n.2 Inputs 5

3.n.3 Expected Outputs & Pass/Fail criteria 5

3.n.4 Test Procedure 5

**4 TEST LOGS 16**

**1 INTRODUCTION**

**1.1 Document Conventions**

|  |  |
| --- | --- |
| **AWS** | Amazon Web Services |

**1.2 System Overview**

This system is composed of a three tier architecture, including a front end user interface component, back end server application component and a database component.

**1.3 Test Approach**

The testing approach that will be taken is one that utilizes an automated testing framework to perform unit tests that verify single component level requirements and system requirements including:

* Unit tests against the back end server application component.
* Unit tests against the database component.
* Integration tests against all system interfaces.

**2 TEST PLAN**

**2.1 Features to be Tested**

All features that are implemented must be tested including:

* User signup
* User login
* User info modifications
* Video uploads
* Video removals
* Video likes
* Video comments
* Find users

**2.2 Features not to be Tested**

Elements that are not being tested include:

* UI Elements
* UI Usability

**2.3 Testing Tools and Environment**

The testing framework that will be used for automated testing is mocha-js. This testing framework allows for unit testing of back application and database components as well as all system interfaces.

**3 TEST CASES**

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-1** | **In File:** test/server\_tests\_001\_user\_signup.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user signup request and responds with the correct json format.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server. 2. Verify that the response is in the correct json format. | | **Inputs**:  {  Username: “TestUser”,  Password: “test\_password”,  Firstname: “Test”,  Lastname: “Test”,  Email: “testuser@test.com”  } | **Outputs:**  Response should be a json object containing properties,  {  message: “”,  result: “”,  User\_Id: “”  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-2** | **In File:** test/server\_tests\_001\_user\_signup.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user signup request and responds with the correct output when a user has been successfully entered into the database.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing valid request data.. 2. Verify that the message property equates to “success” 3. Verify that the User\_Id property equates to a non zero int. | | **Inputs**:  {  Username: “TestUser”,  Password: “test\_password”,  Firstname: “Test”,  Lastname: “Test”,  Email: “testuser@test.com”  } | **Outputs:**  “Message” = “success”  “User\_Id” = (a non zero int) |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-3** | **In File:** test/server\_tests\_001\_user\_signup.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user signup request and responds with the correct output when the inputs are empty strings.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing invalid request data. 2. Verify that the message property in the response is equal to “failure”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  Username: “”,  Password: “”,  Firstname: “”,  Lastname: “”,  Email: “”  } | **Outputs:**  “message” = “failed”  “result” = “invalid request”. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-4** | **In File:** test/server\_tests\_001\_user\_signup.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user signup request and responds with the correct output when no inputs are defined.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing no request data. 2. Verify that the message property in the response is equal to “failure”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  } | **Outputs:**  “message” = “failed”  “result” = “invalid request” |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-5** | **In File:** test/server\_tests\_002\_user\_login.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user login request and responds with the correct json format.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server. 2. Verify that the response is in the correct json format. | | **Inputs**:  {  Username: “Admin”,  Password: “root”  } | **Outputs:**  Response should be a json object containing properties,  {  message: “”,  result: “”,  User\_Id: “”  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-6** | **In File:** test/server\_tests\_002\_user\_login.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user login request and responds with the correct output when a user has entered a successful username and password.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing a successful username and password.. 2. Verify that the message property equates to “success” 3. Verify that the User\_Id property equates to a non zero int. | | **Inputs**:  {  Username: “Admin”,  Password: “root”  } | **Outputs:**  “Message” = “success”  “User\_Id” = (a non zero int) |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-7** | **In File:** test/server\_tests\_002\_user\_login.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user login request and responds with the correct output when a user has entered a successful username and an incorrect password.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing a successful username and incorrect password. 2. Verify that the message property equates to “failed” 3. Verify that the User\_Id property equates to “incorrect password”. | | **Inputs**:  {  Username: “Admin”,  Password: “wrong\_pass”  } | **Outputs:**  “Message” = “failed”  “result” = “incorrect password” |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-8** | **In File:** test/server\_tests\_002\_user\_login.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user login request and responds with the correct output when a user has entered an incorrect username.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing an incorrect username. 2. Verify that the message property equates to “failed” 3. Verify that the result property equates to “incorrect username” | | **Inputs**:  {  Username: “Wrong\_User”,  Password: “root”  } | **Outputs:**  “Message” = “failed”  “result” = “incorrect username” |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-9** | **In File:** test/server\_tests\_002\_user\_login.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user login request and responds with the correct output when the inputs are empty strings.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing invalid request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  Username: “”,  Password: “”  } | **Outputs:**  “message” = “failed”  “result” = “invalid request”. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-10** | **In File:** test/server\_tests\_002\_user\_login.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user login request and responds with the correct output when no inputs are defined.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing no request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  New user credentials,  {  } | **Outputs:**  “message” = “failed”  “result” = “invalid request” |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-11** | **In File:** test/server\_tests\_003\_user\_upload\_video.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user upload video request and responds with the correct json format.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server. 2. Verify that the response is in the correct json format. | | **Inputs**:  {  Video\_Link: “test.com”,  User\_Id: “68”,  Upload\_Date: “2020-05-01T01:01:01.00Z”  } | **Outputs:**  Response should be a json object containing properties,  {  message: “”,  result: “”  Video\_Id: “”  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-12** | **In File:** test/server\_tests\_003\_user\_upload\_video.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user upload video request and responds with the correct output when a user has entered successful video data.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing successful video data. 2. Verify that the message property equates to “success” 3. Verify that the Video\_Id property equates to a non zero int. | | **Inputs**:  {  Video\_Link: “test.com”,  User\_Id: “68”,  Upload\_Date: “2020-05-01T01:01:01.00Z”  } | **Outputs:**  “Message” = “success”  “Video\_Id” = (a non zero int) |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-13** | **In File:** test/server\_tests\_003\_user\_upload\_video.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user upload video request and responds with the correct output when the inputs are empty strings.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing invalid request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  Video\_Link: “”,  User\_Id: “”,  Upload\_Date: “”  } | **Outputs:**  “message” = “failed”  “result” = “invalid request”. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-14** | **In File:** test/server\_tests\_003\_user\_upload\_video.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user upload video data request and responds with the correct output when no inputs are defined.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing no request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  } | **Outputs:**  “message” = “failed”  “result” = “invalid request” |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-15** | **In File:** test/server\_tests\_004\_get\_video\_data.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a get video data request and responds with the correct json format.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server. 2. Verify that the response is in the correct json format. | | **Inputs**:  {  Video\_Id: “68”,  } | **Outputs:**  Response should be a json object containing properties,  {  message: “”,  result: (Dictionary containing all of the video data)  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-16** | **In File:** test/server\_tests\_004\_get\_video\_data.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a get video data request and responds with the correct output when a user has entered a successful Video\_Id.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing successful Video\_Id. 2. Verify that the message property equates to “success” 3. Verify that the “result” property equates to a dictionary object containing all of the video data. | | **Inputs**:  {  Video\_Id: “68”,  } | **Outputs:**  “Message” = “success”  “result” = (Dictionary containing all of the video data) |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-17** | **In File:** test/server\_tests\_004\_get\_video\_data.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a get video data request and responds with the correct output when the inputs are empty strings.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing invalid request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  Video\_Id: “”  } | **Outputs:**  “message” = “failed”  “result” = “invalid request”. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-18** | **In File:** test/server\_tests\_004\_get\_video\_data.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a get video data request and responds with the correct output when no inputs are defined.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing no request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  } | **Outputs:**  “message” = “failed”  “result” = “invalid request” |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-19** | **In File:** test/server\_tests\_005\_get\_all\_user\_video\_data.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a get all user video data request and responds with the correct json format.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server. 2. Verify that the response is in the correct json format. | | **Inputs**:  {  User\_Id: “68”,  } | **Outputs:**  Response should be a json object containing properties,  {  message: “”,  result: (Array containing all of the video data for the user)  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-20** | **In File:** test/server\_tests\_005\_get\_all\_user\_video\_data.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a get all user video data request and responds with the correct output when a user has entered a successful User\_Id..  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing successful User\_Id. 2. Verify that the message property equates to “success” 3. Verify that the “result” property equates to an array containing all of the video data for the user. | | **Inputs**:  {  User\_Id: “68”,  } | **Outputs:**  “Message” = “success”  “result” = (Array containing all of the video data for the user) |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-21** | **In File:** test/server\_tests\_005\_get\_all\_user\_video\_data.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a get all user video data request and responds with the correct output when the inputs are empty strings.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing invalid request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  User\_Id: “”  } | **Outputs:**  “message” = “failed”  “result” = “invalid request”. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-22** | **In File:** test/server\_tests\_005\_get\_all\_user\_video\_data.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a get all user video data request and responds with the correct output when no inputs are defined.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing no request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  } | **Outputs:**  “message” = “failed”  “result” = “invalid request” |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-23** | **In File:** test/server\_tests\_007\_user\_like\_video.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user like video request and responds with the correct json format.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server. 2. Verify that the response is in the correct json format. | | **Inputs**:  {  Video\_Id: “68”,  User\_Id: “68”  } | **Outputs:**  Response should be a json object containing properties,  {  message: “”,  Like\_Id: “”  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-24** | **In File:** test/server\_tests\_007\_user\_like\_video.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user like video request and responds with the correct output when a user has entered a successful Video\_Id and User\_Id.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing successful Video\_Id and User\_Id. 2. Verify that the message property equates to “success” 3. Verify that the “Like\_Id” property equates to a non zero int. | | **Inputs**:  {  Video\_Id: “68”,  User\_Id: “68”  } | **Outputs:**  “Message” = “success”  “Like\_Id” = (a non zero int) |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-25** | **In File:** test/server\_tests\_007\_user\_like\_video.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user like video request and responds with the correct output when the inputs are empty strings.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing invalid request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  Video\_Id: “”,  User\_Id: “”  } | **Outputs:**  “message” = “failed”  “result” = “invalid request”. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-26** | **In File:** test/server\_tests\_007\_user\_like\_video.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user like video request and responds with the correct output when no inputs are defined.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing no request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  } | **Outputs:**  “message” = “failed”  “result” = “invalid request” |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-27** | **In File:** test/server\_tests\_008\_user\_comment\_video.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user comment video request and responds with the correct json format.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server. 2. Verify that the response is in the correct json format. | | **Inputs**:  {  Comment: “test comment”,  Video\_Id: “68”,  User\_Id: “68”  } | **Outputs:**  Response should be a json object containing properties,  {  message: “”,  Comment\_Id: “”  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-28** | **In File:** test/server\_tests\_008\_user\_comment\_video.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user comment video request and responds with the correct output when a user has entered a successful Comment, Video\_Id and User\_Id.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing successful Comment, Video\_Id and User\_Id. 2. Verify that the message property equates to “success” 3. Verify that the “Comment\_Id” property equates to a non zero int. | | **Inputs**:  {  Comment: “test comment”  Video\_Id: “68”,  User\_Id: “68”  } | **Outputs:**  “Message” = “success”  “Comment\_Id” = (a non zero int) |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-29** | **In File:** test/server\_tests\_008\_user\_comment\_video.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user comment video request and responds with the correct output when the inputs are empty strings.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing invalid request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  Comment: “”,  Video\_Id: “”,  User\_Id: “”  } | **Outputs:**  “message” = “failed”  “result” = “invalid request”. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-30** | **In File:** test/server\_tests\_008\_user\_comment\_video.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user comment video request and responds with the correct output when no inputs are defined.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing no request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  } | **Outputs:**  “message” = “failed”  “result” = “invalid request” |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-31** | **In File:** test/server\_tests\_009\_user\_video\_likes.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user video likes request and responds with the correct json format.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server. 2. Verify that the response is in the correct json format. | | **Inputs**:  {  Video\_Id: “68”  } | **Outputs:**  Response should be a json object containing properties,  {  message: “”,  result: “”  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-32** | **In File:** test/server\_tests\_009\_user\_video\_likes.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user video likes request and responds with the correct output when a user has entered a successful Video\_Id.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing successful Video\_Id and User\_Id. 2. Verify that the message property equates to “success” 3. Verify that the “result” property equates to an array containing all likes for the specified video”. | | **Inputs**:  {  Video\_Id: “68”  } | **Outputs:**  “Message” = “success”  “result” = (Array containing all likes for the specified video) |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-33** | **In File:** test/server\_tests\_009\_user\_video\_likes.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user video likes request and responds with the correct output when the inputs are empty strings.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing invalid request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  Video\_Id: “”  } | **Outputs:**  “message” = “failed”  “result” = “invalid request”. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-34** | **In File:** test/server\_tests\_009\_user\_video\_likes.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user video likes request and responds with the correct output when no inputs are defined.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing no request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  } | **Outputs:**  “message” = “failed”  “result” = “invalid request” |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-35** | **In File:** test/server\_tests\_010\_user\_video\_comments.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user video comments request and responds with the correct json format.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server. 2. Verify that the response is in the correct json format. | | **Inputs**:  {  Video\_Id: “68”  } | **Outputs:**  Response should be a json object containing properties,  {  message: “”,  result: “”  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-36** | **In File:** test/server\_tests\_010\_user\_video\_comments.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user video comments request and responds with the correct output when a user has entered a successful Video\_Id.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing successful Video\_Id. 2. Verify that the message property equates to “success” 3. Verify that the “result” property equates to an array containing all comments for the specified video”. | | **Inputs**:  {  Video\_Id: “68”  } | **Outputs:**  “Message” = “success”  “result” = (Array containing all comments for the specified video) |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-37** | **In File:** test/server\_tests\_010\_user\_video\_comments.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user video comments request and responds with the correct output when the inputs are empty strings.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing invalid request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  Video\_Id: “”  } | **Outputs:**  “message” = “failed”  “result” = “invalid request”. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-38** | **In File:** test/server\_tests\_010\_user\_video\_comments.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user video comments request and responds with the correct output when no inputs are defined.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing no request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  } | **Outputs:**  “message” = “failed”  “result” = “invalid request” |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-39** | **In File:** test/server\_tests\_011\_user\_delete\_video.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user delete video request and responds with the correct json format.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server. 2. Verify that the response is in the correct json format. | | **Inputs**:  {  Video\_Id: “68”  } | **Outputs:**  Response should be a json object containing properties,  {  message: “”,  } |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-40** | **In File:** test/server\_tests\_011\_user\_delete\_video.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user delete video request and responds with the correct output when a user has entered a successful Video\_Id.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing successful Video\_Id. 2. Verify that the message property equates to “success” | | **Inputs**:  {  Video\_Id: “68”  } | **Outputs:**  “message” = “success” |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-41** | **In File:** test/server\_tests\_011\_user\_delete\_video.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user delete video request and responds with the correct output when the inputs are empty strings.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing invalid request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  Video\_Id: “”  } | **Outputs:**  “message” = “failed”  “result” = “invalid request”. |

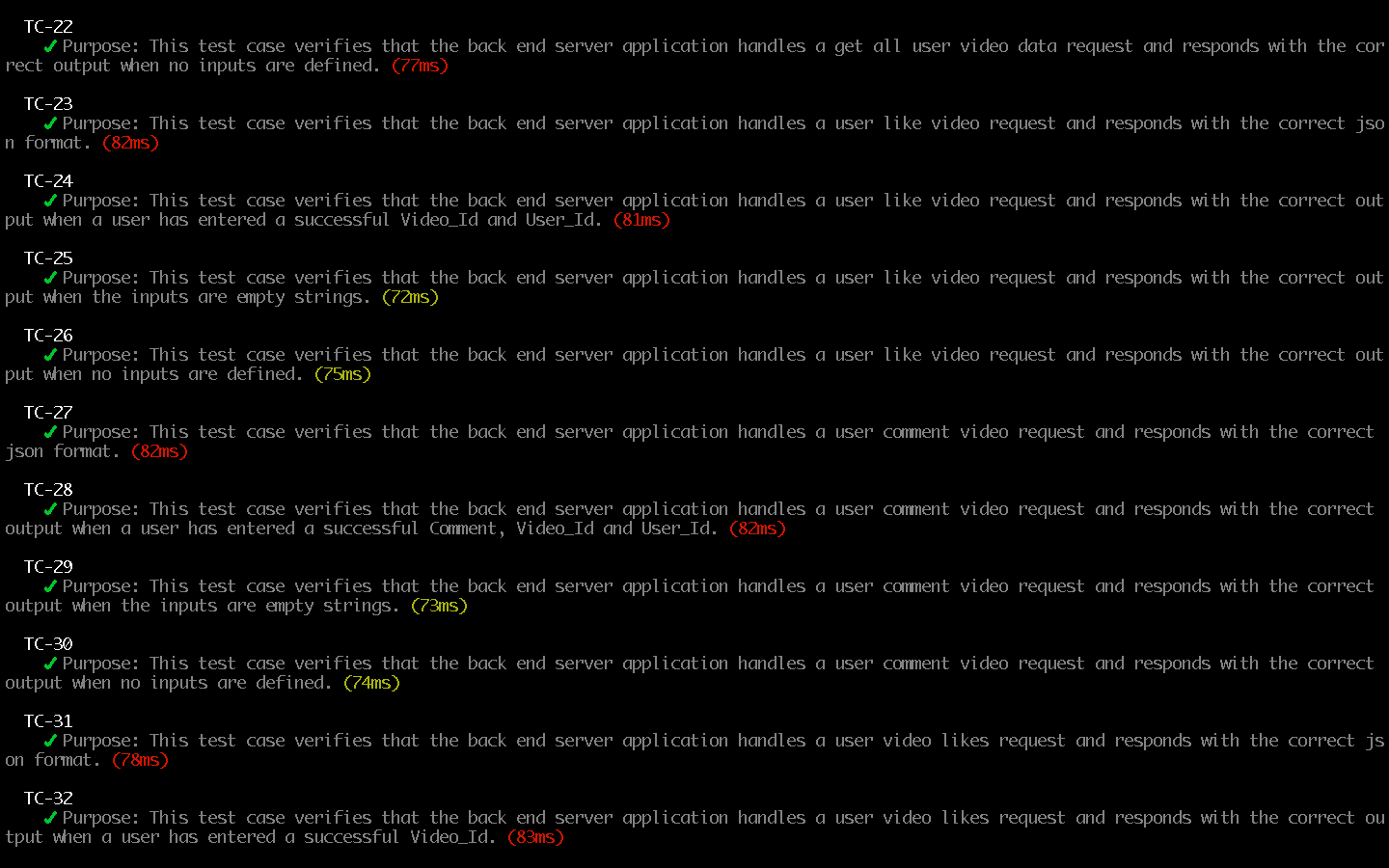
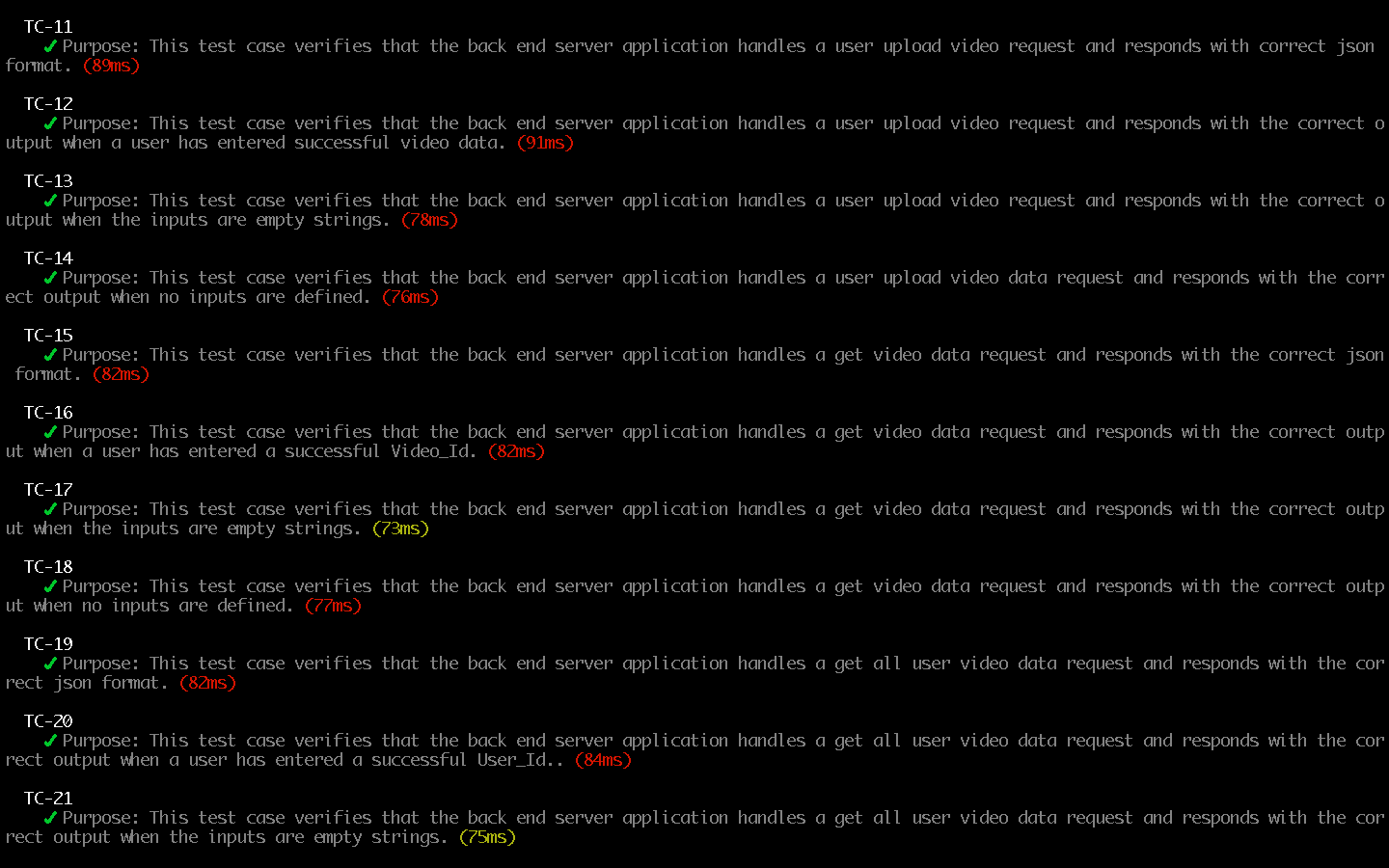
|  |  |  |  |
| --- | --- | --- | --- |
| **TC-42** | **In File:** test/server\_tests\_011\_user\_delete\_video.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a user video comments request and responds with the correct output when no inputs are defined.  **Test Procedure:**   1. Test sends HTTP POST request to back end application server containing no request data. 2. Verify that the message property in the response is equal to “failed”. 3. Verify that the result property in the response is equal to “invalid request”. | | **Inputs**:  {  } | **Outputs:**  “message” = “failed”  “result” = “invalid request” |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC-43** | **In File:** test/server\_tests\_012\_find\_users.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a find users request and responds with the correct json format.  **Test Procedure:**   1. Test sends HTTP GET request to back end application server. 2. Verify that the response is in the correct json format. | | **Inputs**:  N/A | **Outputs:**  Response should be a json object containing properties,  {  message: “”,  result: “”  } |

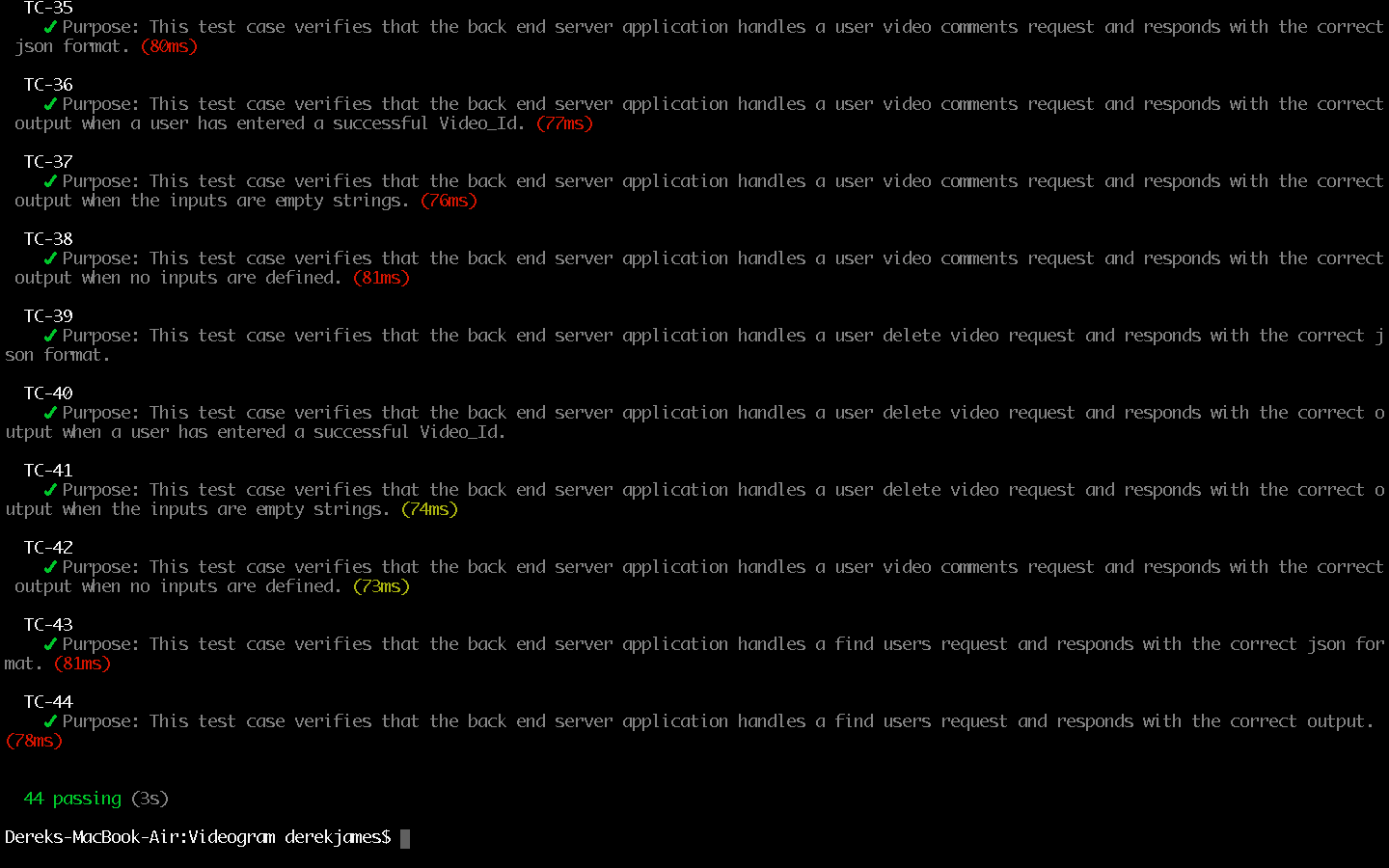
|  |  |  |  |
| --- | --- | --- | --- |
| **TC-44** | **In File:** test/server\_tests\_012\_find\_users.js | | |
| **Purpose**:  This test case verifies that the back end server application handles a find users request and responds with the correct output.  **Test Procedure:**   1. Test sends HTTP GET request to back end application server. 2. Verify that the message property equates to “success” 3. Verify that the result property equates to an Array of users | | **Inputs**:  N/A | **Outputs:**  “message” = “success”  “result”: “(Array of users)” |

**4 TEST LOGS**

**Note: No logs available, (Includes Screenshot result of all test cases being run).**

****

****

****