

# Verification and Validation Report: Software Engineering

Team #11, OKKM Insights

Mathew Petronilho

Oleg Glotov

Kyle McMaster

Kartik Chaudhari

March 10, 2025

# 1 Revision History

| Date   | Version | Notes |
|--------|---------|-------|
| Date 1 | 1.0     | Notes |
| Date 2 | 1.1     | Notes |

## 2 Symbols, Abbreviations and Acronyms

| symbol | description |
|--------|-------------|
| T      | Test        |

[symbols, abbreviations or acronyms – you can reference the SRS tables if needed —SS]

# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Revision History</b>                      | <b>i</b>  |
| <b>2</b> | <b>Symbols, Abbreviations and Acronyms</b>   | <b>ii</b> |
| <b>3</b> | <b>Functional Requirements Evaluation</b>    | <b>1</b>  |
| 3.1      | Summary of Tests . . . . .                   | 1         |
| 3.2      | Analysis . . . . .                           | 9         |
| <b>4</b> | <b>Nonfunctional Requirements Evaluation</b> | <b>10</b> |
| 4.1      | Look and Feel . . . . .                      | 10        |
| 4.1.1    | Summary of Tests . . . . .                   | 10        |
| 4.1.2    | Analysis . . . . .                           | 11        |
| 4.2      | Usability . . . . .                          | 11        |
| 4.3      | Performance . . . . .                        | 11        |
| 4.4      | Operational and Environmental . . . . .      | 11        |
| 4.4.1    | Summary of Tests . . . . .                   | 11        |
| 4.4.2    | Analysis . . . . .                           | 15        |
| 4.5      | Maintainability and Support . . . . .        | 15        |
| 4.5.1    | Summary of Tests . . . . .                   | 15        |
| 4.5.2    | Analysis . . . . .                           | 15        |
| 4.6      | Security . . . . .                           | 16        |
| 4.6.1    | Summary of Tests . . . . .                   | 16        |
| 4.6.2    | Analysis . . . . .                           | 19        |
| 4.7      | Cultural . . . . .                           | 19        |
| 4.8      | Compliance . . . . .                         | 19        |
| 4.9      | User Documentation and Training . . . . .    | 19        |
| <b>5</b> | <b>Comparison to Existing Implementation</b> | <b>19</b> |
| <b>6</b> | <b>Unit Testing</b>                          | <b>20</b> |
| 6.1      | Front-end . . . . .                          | 20        |
| 6.1.1    | Rendering of a Component . . . . .           | 20        |
| 6.1.2    | Open Pop Up . . . . .                        | 20        |
| 6.1.3    | Close Pop Up . . . . .                       | 20        |
| 6.1.4    | Header Logged In . . . . .                   | 21        |
| 6.1.5    | Header Logged Out . . . . .                  | 21        |
| 6.1.6    | Header Re-directions . . . . .               | 21        |

|           |  |           |
|-----------|--|-----------|
| 6.1.7     | Login Success . . . . .                        | 22        |
| 6.1.8     | Login Fail . . . . .                           | 22        |
| 6.1.9     | New Project Validation . . . . .               | 22        |
| 6.1.10    | New Project Creation Success . . . . .         | 22        |
| 6.1.11    | New Project Creation Failure . . . . .         | 23        |
| 6.1.12    | Project Section renders all projects . . . . . | 23        |
| 6.1.13    | Project Tile Navigation . . . . .              | 23        |
| 6.1.14    | Register Dynamic Password Validation . . . . . | 23        |
| 6.1.15    | Register Success . . . . .                     | 24        |
| 6.1.16    | Register Fail . . . . .                        | 24        |
| 6.1.17    | Update Info Success . . . . .                  | 24        |
| 6.1.18    | Update Info Fail . . . . .                     | 24        |
| <b>7</b>  | <b>Changes Due to Testing</b>                  | <b>25</b> |
| 7.1       | Changes to Front-end . . . . .                 | 25        |
| <b>8</b>  | <b>Automated Testing</b>                       | <b>26</b> |
| <b>9</b>  | <b>Trace to Requirements</b>                   | <b>26</b> |
| <b>10</b> | <b>Trace to Modules</b>                        | <b>26</b> |
| <b>11</b> | <b>Code Coverage Metrics</b>                   | <b>26</b> |
| 11.1      | Front-end Coverage . . . . .                   | 26        |

## List of Tables

## List of Figures

|   |   |    |
|---|---|----|
| 1 | Front-end Unit Testing Coverage Results . . . . . | 27 |
|---|---|----|

This document ...

## 3 Functional Requirements Evaluation

### 3.1 Summary of Tests

| Test ID | Status | Notes  |
|---------|--------|--|
| T-FR0   | Pass   |  |
| T-FR1   | Pass   |  |
| T-FR2   | Pass   | Not all fields from test are in user information |
| T-FR3   | N/A    | Requirement covered is out of scope              |
| T-FR4   | Pass   | Confirmed Payment no longer required             |
| T-FR5   | Pass   | Download not yet supported                       |
| T-FR6   | Fail   | Large image files fail to upload                 |
| T-FR7   | Fail   | Not yet implemented                              |
| T-FR8   | Fail   | Not yet implemented                              |
| T-FR9   | Pass   |  |
| T-FR10  | Pass   |  |
| T-FR11  | Pass   |  |
| T-FR12  | N/A    | Requirement covered is out of scope              |
| T-FR13  | Pass   |  |
| T-FR14  | Fail   | Not yet implemented                              |

#### T-FR0: Customer Account Creation

1. **Initial State:** The system is accessible, and no user account currently exists for the test user.
2. **Input:**
  - Customer provides valid personal information:
    - **Name:** “Alice Smith”
    - **Email:** `alice.smith@example.com`
    - **Password:** “StrongPassword!2021”

- Customer agrees to the system's privacy policy.

**3. Expected Result:**

- A new user account is created.
- Account information is securely stored in the database.
- The customer is redirected to the login page with a success message.

**4. Result:** Pass

**T-FR1: Customer Authentication**

**1. Initial State:** An existing user account with the following credentials:

- **Email:** `user.test@example.com`
- **Password:** "TestPass#123"

**2. Input:**

- Customer enters the correct login credentials:
  - **Email:** `user.test@example.com`
  - **Password:** "TestPass#123"

**3. Expected Result:**

- Customer is successfully authenticated.
- Access to privileged information (e.g., user dashboard) is granted.
- A session token or cookie is established for the user session.

**4. Result:** Pass

**T-FR2: Customer Account Modification**

**1. Initial State:** Customer is logged in and has access to their account information page.

**2. Input:**

- Customer updates personal information fields:

- **Address:** “123 New Street, Cityville”
- **Phone Number:** “555-1234”
- **Profile Picture:** Uploads a new image file.

3. **Expected Result:**

- Updated personal information is saved and reflected in the database.
- The user receives a confirmation message indicating successful update.

4. **Result:** Pass

5. **Comments:** Not all fields are used in customer profile. However, functionality remains testable and validated.

### **T-FR3: Payment Processing**

1. **Initial State:** Customer is logged in and ready to make a payment for a service request.

2. **Input:**

- Valid payment information:
  - **Credit Card Number:** “4111 1111 1111 1111” (Test Visa number)
  - **Expiry Date:** “12/25”
  - **CVV:** “123”
  - **Billing Address:** Matches the address on file.

3. **Expected Result:**

- Payment is processed successfully.
- A confirmation receipt is generated and emailed to the customer.
- The service request status is updated to “Paid” or equivalent.

4. **Result:** Test not completed

5. **Comments:** Requirement is out of scope



#### **T-FR4: Service Request Submission**

1. **Initial State:** Customer is logged in and has a confirmed payment.
2. **Input:**
  - Customer fills out the service request form with necessary details:
    - **Service Type:** “Image Analysis”
    - **Description:** “Analysis of satellite images for deforestation.”
    - **Preferred Completion Date:** “2023-12-31”
3. **Expected Result:**
  - Service request is accepted and logged in the system.
  - Customer receives a confirmation message and request ID.
4. **Result:** Pass
5. **Comments:** Payment is no longer required in initial state.

#### **T-FR5: Service Report Delivery**

1. **Initial State:** Customer is logged in and has a completed service request.
2. **Input:**
  - Customer navigates to the “My Reports” section after being notified of service completion.
3. **Expected Result:**
  - The service report is available for viewing and download.
  - Report contents are accurate and correspond to the service request.
4. **Result:** Pass
5. **Comments:** Download not yet supported

### **T-FR6: Image Upload**

1. **Initial State:** Customer is logged in with an active service request requiring image uploads.
2. **Input:**
  - Customer uploads multiple image files:
    - Image1.jpg: 2 MB
    - Image2.png: 3 MB
    - Image3.tif: 5 MB
3. **Expected Result:**
  - All images are successfully uploaded and stored.
  - Images are correctly linked to the specific service request.
  - Customer receives an upload success message.
4. **Result:** Fail
5. **Comments:** Large image files do not upload

### **T-FR7: Satellite Image Request**

1. **Initial State:** Customer is logged in with an active service request that requires satellite images.
2. **Input:**
  - Geographic coordinates:
    - **Latitude:** 37.7749° N
    - **Longitude:** 122.4194° W (San Francisco, CA)
    - **Date Range:** “2023-01-01” to “2023-01-31”
3. **Expected Result:**
  - The system retrieves and stores satellite images corresponding to the provided coordinates and date range.
  - Customer is notified of successful retrieval.

4. **Result:** Fail
5. **Comments:** Not yet implemented

#### **T-FR8: Service Request Failure Alert**

1. **Initial State:** Customer has initiated a service request that cannot be fulfilled due to invalid parameters.
2. **Input:**
  - Service request with unfulfillable criteria:
    - **Service Type:** “Image Analysis”
    - **Geographic Coordinates:** Invalid coordinates (e.g., Latitude: 95° N)
3. **Expected Result:**
  - Customer receives an alert indicating that the service request cannot be processed.
  - An explanation of the failure is provided.
4. **Result:** Fail
5. **Comments:** Not yet implemented

#### **T-FR9: Labeler Account Creation**

1. **Initial State:** No labeler account exists for the test user in the system.
2. **Input:**
  - Labeler provides required account information:
    - **Name:** “Bob Labeler”
    - **Email:** bob.labeler@example.com
    - **Password:** “LabelerPass789!”
    - **Expertise Area:** “Satellite Image Annotation”
3. **Expected Result:**

- A new labeler account is created and securely stored.
- Labeler is prompted to complete any additional on-boarding steps.

4. **Result:** Pass

#### **T-FR10: Labeler Authentication**

1. **Initial State:** Labeler account exists with credentials:

- **Email:** `labeler.test@example.com`
- **Password:** “LabelerSecure!2022”

2. **Input:**

- Labeler enters correct login credentials.

3. **Expected Result:**

- Labeler is authenticated successfully.
- Access to the labeler dashboard is granted.

4. **Result:** Pass

#### **T-FR11: Labeler Account Modification**

1. **Initial State:** Labeler is logged in and on the account settings page.

2. **Input:**

- Update personal information:
  - **Expertise Area:** Add “Aerial Photography Annotation”
  - **Contact Number:** “555-6789”

3. **Expected Result:**

- Personal information is updated and stored in the database.
- Labeler receives a confirmation of successful update.

4. **Result:** Pass

## FR12: Labeler Earnings Transfer Test

1. **Initial State:** Labeler is logged in with available earnings exceeding the minimum transfer threshold.
2. **Input:**
  - Transfer request to linked banking platform:
    - **Amount:** Total available earnings.
    - **Bank Account Details:** Pre-verified and linked.
3. **Expected Result:**
  - Earnings are transferred successfully.
  - Transaction record is created.
  - Labeler receives confirmation and updated earnings balance.
4. **Result:** Test not completed
5. **Comments:** Requirement is out of scope

## T-FR13: Image Annotation

1. **Initial State:** Labeler is logged in with images assigned for annotation.
2. **Input:**
  - Labeler annotates an image using the provided tools:
    - Draws bounding boxes around objects.
    - Adds classification labels to each object.
    - Saves the annotation.
3. **Expected Result:**
  - Annotated image is stored in the system.
  - Annotation data is correctly linked to the image and service request.
  - Labeler receives confirmation of successful submission.
4. **Result:** Pass

### **T-FR14: Consolidated Annotation Report**

1. **Initial State:** All required labeler annotations are complete for a service request.
2. **Input:**
  - System triggers consolidation process for annotations.
3. **Expected Result:**
  - Consolidated report is generated if label accuracy meets the pre-defined threshold.
  - Report is stored and made accessible to the customer.
4. **Result:** Fail
5. **Comments:** Not yet implemented.

## **3.2 Analysis**

Overall, the validation team is happy with the progress towards satisfying our refined functional requirements. Of the 13 attempted tests, nine were successful. Of the four which were unsuccessful, only one failed due to some error related to implementation. The others are expected to fail, as they have yet to be implemented. The unexpected failure is ‘T-FR6’, which has showed that large image files are unable to be uploaded to the project creation portal. The team has been able to trace this error back to our web hosting platform, and are in the process of identifying potential fixes. After this fix, and the implementation of report creation and satellite image requests, we will have demonstrated 100% functional requirement validation.

## 4 Nonfunctional Requirements Evaluation

### 4.1 Look and Feel

#### 4.1.1 Summary of Tests

| Test ID | Status | Notes |
|---------|--------|-------|
| T-LF0   | Pass   |       |
| T-LF1   | Pass   |       |
| T-LF2   | Pass   |       |

##### 1. T-LF0: Responsive Layout Validation

**Initial State:** Users access the application on devices with screen resolutions ranging from 1024×768 pixels to 1920×1080 pixels.

**Input/Condition:** The application is displayed on various screen sizes within the specified range to verify adaptability and layout consistency.

**Expected Result:** All elements are displayed on the screen, regardless of dimensions.

**Result** Pass

##### 2. T-LF1: Interactive Elements Feedback Validation

**Initial State:** Interactive elements (e.g., buttons, links) are present within the application interface.

**Input/Condition:** Users interact with various interactive elements to verify that visual feedback is provided appropriately.

**Expected Result:** Each interactive element provides feedback when hovered over and clicked on.

**Result** Pass

##### 3. T-LF2: Unified Visual Design Validation

**Initial State:** The application interface displays all components (buttons, menus, text fields, images, etc.) with the unified visual design specifications.

**Input/Condition:** Users or testers navigate through the application to assess the consistency of font type, sizing, color, and background tones across all components.

**Expected Result:** All elements follow consistent style

**Result** Pass

**Comments** Replacing Label Studio with our own framework was necessary to pass this test.

#### 4.1.2 Analysis

All three of the Look and Feel tests pass. These tests were designed to provide complete coverage of the Look and Feel requirements, and therefore we believe them to be covered.

## 4.2 Usability

Please see usability report, found [here](#)

## 4.3 Performance

## 4.4 Operational and Environmental

### 4.4.1 Summary of Tests

| Test ID | Status | Notes                       |
|---------|--------|-----------------------------|
| T-OE0   | N/A    | Requirement is out of scope |
| T-OE1   | Fail   | Not yet implemented         |
| T-OE2   | N/A    | Requirement is out of scope |
| T-OE3   | N/A    | Requirement is out of scope |
| T-OE4   | Fail   | Not yet implemented         |
| T-OE5   | Fail   | Not yet implemented         |
| T-OE6   | Pass   |                             |
| T-OE7   | Pass   |                             |
| T-OE8   | Pass   |                             |
| T-OE9   | Pass   |                             |



1. T-OE0: Energy Efficiency Validation

**Initial State:** The system is running on cloud infrastructure with existing server management configurations.

**Input/Condition:** Implement energy-efficient practices in cloud usage and server management, then measure energy consumption before and after optimization.

**Expected Result:** A statistically significant decrease in energy usage with 5% sensitivity.

**Result:** Test not completed

**Comments:** Requirement is out of scope

2. T-OE1: API and Data Format Integration Validation

**Initial State:** The system is configured with API access credentials for at least two major satellite data providers.

**Input/Condition:** The system attempts to automatically acquire and integrate satellite images from the specified providers using their standardized APIs and data formats.

**Expected Result:** System obtains result from API provider.

**Result:** Fail

**Comments:** Not yet implemented

3. T-OE2: Payment Processor Integration Validation

**Initial State:** The system is configured with API access credentials for reliable and secure payment processors (e.g., Stripe, PayPal).

**Input/Condition:** Users and clients perform financial transactions through the integrated payment gateways.

**Expected Result:** System successfully processes payment

**Result:** Test not completed

**Comments:** Requirement is out of scope

4. T-OE3: Multiple Currency Support Validation

**Initial State:** The system is configured to support multiple currencies, including USD, EUR, GBP, and INR.

**Input/Condition:** Users perform transactions in each supported currency to verify correct processing.

**Expected Result:** System successfully processes payment with all valid currencies

**Result:** Test not completed

**Comments:** Requirement is out of scope

5. T-OE4: Machine Learning Framework Compatibility Validation

**Initial State:** The system is configured with machine learning frameworks such as TensorFlow, PyTorch, and scikit-learn installed.

**Input/Condition:** Users train and deploy models using each framework to verify compatibility.

**Expected Result:** System successfully deploys models to each framework

**Result:** Fail

**Comments:** Not yet implemented

6. T-OE5: Data Pipeline Efficiency Validation

**Initial State:** The system has established data pipelines for transferring labeled datasets between the platform and ML models.

**Input/Condition:** Large labeled datasets (e.g., 10,000 images) are transferred through the data pipelines.

**Expected Result:** Data pipeline can support large datasets

**Result:** Fail

**Comments:** Not yet implemented

7. T-OE6: Web Browser Accessibility Validation

**Initial State:** Users have access to the platform's web URL.

**Input/Condition:** Users attempt to access and use the platform via various supported web browsers without installing any software.

**Expected Results:** Users are able to complete all key actions

**Results:** Pass

8. T-OE7: Road Map Consistency

**Initial State:** Application has a release road map that is publicly accessible.

**Input/Condition:** Team member conducts a review.

**Expected Results:** At least MIN\_ON\_TIME\_MILESTONE% of the listed milestones have been met on time.

**Result:** Pass

9. Beta Testing: T-OE8

**Initial State:** Beta version of application is deployed and accessible

**Input/Condition:** At least BETA\_TESTERS beta testers are provided access to use the application.

**Expected Results:** Feedback on any bugs, navigation issues, or aesthetic problems is provided. Less than MAX\_BUGS\_FOUND bugs are found.

**Result:** Pass

**Comments:** Preliminary usability testing has been performed. See usability testing for details.

10. Regression Testing: T-OE9

**Initial State:** Application is deployed.

**Input/Condition:** Run regression test suite, consisting of unit tests.

**Expected Results:** All regression tests are passed.

**Result:** Pass

#### 4.4.2 Analysis

When conducting the validation of these requirements, we have identified a weakness of our current implementation. Of the seven attempted tests, only four passed. Of the ones that failed, each one is due to a lack of focus from the development team. Now that there is a core structure in the application, we are able to begin addressing additional features, such as high performance data pipelines.

### 4.5 Maintainability and Support

#### 4.5.1 Summary of Tests

| Test ID | Status | Notes   |
|---------|--------|---|
| T-MS0   | N/A    | Test not yet attempted due to project being in early development. |

1. T-MS0: Ease of Change

**Initial State:** Application's source repository contains complete documentation.

**Input/Condition:** Competent software developer who has not previously worked on the app reviews documentation and attempts to perform tasks.

**Expected Results:** The developer can easily make a minor update to a specified part of the application.

**Results:** Test not completed

**Comments:** Test not yet attempted due to project being in early development.

#### 4.5.2 Analysis

Although this test has not been attempted, the development team has been careful to write code that will be maintainable and match the design doc-

ument description. This will ensure when this test is attempted, it will be successful.

## 4.6 Security

### 4.6.1 Summary of Tests

| Test ID | Status | Notes                           |
|---------|--------|---------------------------------|
| T-SE0   | Pass   |                                 |
| T-SE1   | Pass   |                                 |
| T-SE2   | Pass   |                                 |
| T-SE3   | Pass   |                                 |
| T-SE4   | Fail   | Some passwords incorrectly fail |
| T-SE5   | Pass   | Not yet implemented             |
| T-SE6   | Fail   |                                 |
| T-SE7   | Fail   |                                 |
| T-SE8   | N/A    | Requirement is out of scope     |
| T-SE9   | Pass   |                                 |

#### 1. T-SE0: Logged Out Permissions

**Initial State:** Application is deployed.

**Input/Condition:** Tester who is not signed in tries to access application paths for project creation and image labeling (Ex. /projects or /label).

**Expected Results:** The tester is denied access to these paths and is told to sign in.

**Result:** Pass

#### 2. T-SE1: Labeler Permissions

**Initial State:** Application is deployed.

**Input/Condition:** Tester who is signed in as a labeler tries to access application paths for project creation.

**Expected Results:** The tester is denied access to these paths. However, the tester has access to paths related to image labeling.

**Result:** Pass

3. T-SE2: Invalid Email Format

**Initial State:** Front-end registration page is created and integrated with the database.

**Input/Condition:** Email with invalid format, such as an empty string or a string missing '@', is entered.

**Expected Results:** Application rejects email and tells the user that the email format is wrong.

**Result:** Pass

4. T-SE3: Duplicate Email

**Initial State:** Front-end registration page is created and integrated with the database.

**Input/Condition:** Email that is already in database is entered.

**Expected Results:** Application rejects email and tells the user that the email is in use.

**Result:** Pass

5. T-SE4: Invalid Password Format

**Initial State:** Front-end registration page is created and integrated with the database.

**Input/Condition:** Password with invalid format, such as an empty string or a string with no numbers, is entered.

**Expected Results:** Application rejects password and tells the user what requirements they have not met.

**Result:** Fail

**Comments:** Some passwords which should pass, such as 'Password123#' fail.

6. T-SE5: System Error

**Initial State:** Application is deployed.

**Input/Condition:** Purposely invoke a system failure, and attempt to perform an action such as a label submission.

**Expected Results:** Application provides an error message on the user interface. The database has not changed in anyway.

**Result:** Pass

7. T-SE6: Duplicate Entries

**Initial State:** Database is deployed.

**Input/Condition:** Duplicate database entry is inserted into the database.

**Expected Results:** Database has only one of the inputted entry and the duplicate has been removed.

**Result:** Fail

**Comments:** Not yet implemented

8. T-SE7: Encrypted User Data

**Initial State:** Application is deployed.

**Input/Condition:** Tester registers an account.

**Expected Results:** All sensitive user data that is stored in the database is encrypted.

**Result:** Fail

**Comments:** Not yet implemented

9. T-SE8: Encrypted Payments

**Initial State:** Application is deployed.

**Input/Condition:** Tester enters sample payment details to pay for a labeling project that has been created.

**Expected Results:** These details are encrypted and can not be read through packet analyzers. The amount in the request can not be modified by an adversary.

**Result:** Test not completed

**Comments:** Requirement is out of scope

#### 10. T-SE9: SQL Injection

**Initial State:** Application is deployed.

**Input/Condition:** A malicious SQL statement is entered into a text field.

**Expected Results:** The system raises an error telling the user that it is invalid.

**Result:** Pass

#### 4.6.2 Analysis

This application has been designed for security, and that is clear from the tests results. Six of the nine security tests pass, with partial passes for two of the failing tests. For the encryption test (T-SE7), user data is not yet encrypted, but passwords are. For the password validation test (T-SE4), all weak passwords are blocked, but some strong passwords are as well. The development team will address this issue in coming releases, but we are glad to know that if anything, we are not overly permissive. The requirement related to the remaining failed test (T-SE6), has been deemed ‘Medium’ priority, and may be addressed in a future release.

### 4.7 Cultural

### 4.8 Compliance

### 4.9 User Documentation and Training

## 5 Comparison to Existing Implementation

This section will not be appropriate for every project.



## 6 Unit Testing

### 6.1 Front-end

Please refer to the tests folder in the frontend directory found [here](#).

#### 6.1.1 Rendering of a Component

- Description: A unit test was written for each component to ensure that it renders without error
- Inputs: The component
- Expected Outputs: The component renders
- Result: Pass

#### 6.1.2 Open Pop Up

- Description: A unit test was written for each pop up component to ensure the pop up appears when open
- Inputs: `open := true`
- Expected Outputs: The pop up renders
- Result: Pass

#### 6.1.3 Close Pop Up

- Description: A unit test was written for each pop up component to ensure the pop up does not appear when closed
- Inputs: `open := false`
- Expected Outputs: The pop up does not render
- Result: Pass

#### **6.1.4 Header Logged In**

- Description: Ensure the header renders the right things when the user is logged in
- Inputs: logged in := true
- Expected Outputs: The header should contain the log out button and profile button
- Result: Pass

#### **6.1.5 Header Logged Out**

- Description: Ensure the header renders the right things when the user is logged out
- Inputs: logged in := false
- Expected Outputs: The header should contain the log in button and register button
- Result: Pass

#### **6.1.6 Header Re-directions**

- Description: Ensure the headers buttons redirect to the expected link
- Inputs: N/A
- Expected Outputs: The header redirects to the login on pressing the login button, register on pressing the register button, home when clicking the logout button, and edit profile information when clicking the profile button
- Result: Pass

#### **6.1.7 Login Success**

- Description: Ensure the authorization context is set up upon successful login
- Inputs: valid email and password
- Expected Outputs: Login is successful and authorization context is set up
- Result: Pass

#### **6.1.8 Login Fail**

- Description: Ensure error message is displayed on login fail
- Inputs: invalid email and password
- Expected Outputs: Message saying invalid credentials
- Result: Pass

#### **6.1.9 New Project Validation**

- Description: Ensure required inputs are filled and notify the user if not
- Inputs: empty required fields such as name
- Expected Outputs: Message saying what required fields have not been filled in
- Result: Pass

#### **6.1.10 New Project Creation Success**

- Description: Ensure form submission occurs and success pop up is activated when server creates project
- Inputs: Entirely filled out project creation form
- Expected Outputs: Success pop up shown
- Result: Pass

#### **6.1.11 New Project Creation Failure**

- Description: Ensure form submission occurs and failure pop up is activated when a server side error occurs
- Inputs: Entirely filled out project creation form
- Expected Outputs: Failure pop up shown
- Result: Pass

#### **6.1.12 Project Section renders all projects**

- Description: Ensure projects section component renders all projects given to it
- Inputs: A list of projects
- Expected Outputs: Each project has its own project card on the page
- Result: Pass

#### **6.1.13 Project Tile Navigation**

- Description: Ensure project tile redirects to the correct page
- Inputs: tile type
- Expected Outputs: When the tile type is label, it redirects to the label project. When the tile type is client, it redirects to project insights.
- Result: Pass

#### **6.1.14 Register Dynamic Password Validation**

- Description: Ensure the password conditions show as satisfied when given a valid password
- Inputs: A valid password
- Expected Outputs: Password conditions show as satisfied
- Result: Pass

#### **6.1.15 Register Success**

- Description: Ensure the form is submitted, shows a success pop up, and redirect
- Inputs: valid email and password
- Expected Outputs: Success pop up comes up and redirected to the login page
- Result: Pass

#### **6.1.16 Register Fail**

- Description: Ensure the user is notified if the account already exists
- Inputs: duplicate email
- Expected Outputs: Message saying the account already exists
- Result: Pass

#### **6.1.17 Update Info Success**

- Description: Ensure the form is submitted and the new information is now displayed
- Inputs: valid email change
- Expected Outputs: Email on the account information page is updated to the new email
- Result: Pass

#### **6.1.18 Update Info Fail**

- Description: Ensure the user is notified if the account already exists, do not allow update
- Inputs: duplicate email
- Expected Outputs: Message saying the account already exists
- Result: Pass

## 7 Changes Due to Testing

[This section should highlight how feedback from the users and from the supervisor (when one exists) shaped the final product. In particular the feedback from the Rev 0 demo to the supervisor (or to potential users) should be highlighted. —SS]

### 7.1 Changes to Front-end

Our labeling tool was largely refactored to incorporate the feedback we received from our usability testing. We also considered some of the unit testing outcomes. These changes included:

- Added clearer visual feedback to all the buttons present in the labeling tool. Also made the currently selected label type more obvious to the user.
- Changed the contextual pop ups to include more detailed descriptions and any short cuts associated with a button.
- Added more details and made steps more granular in the help walk-through of the labeling tool. These additional details should help the user in further understanding what they need to do.
- Changed the text of the main submission buttons so that it was clear what would happen when they were pressed. For example, submit was renamed to "submit labels".
- Fixed a bug where the tool would get stuck if the submit button was pushed when there was no labels made.
- The label button now stays selected after a label is created so the user can seamlessly label multiple objects of the same class without having to reselect it every time.
- A visual gif will be added to show the basic process of creating a label so that it is clear the labels are to be drawn on the image.
- Rather than have tools spread out, they have all been condensed into an easy access toolbar.

- New button was added to reset zoom, contrast, brightness and image position back to its initial state.
- Removed white space.
- Removed help button when the project was complete.

## 8 Automated Testing

Automated testing and linting have been implemented in the frontend and backend repositories, as described in section 7.2 of the [Development Plan](#).

## 9 Trace to Requirements

The traceability from tests to requirements can be seen in section 4.3 of the [VnV Plan](#).

## 10 Trace to Modules

The traceability from requirements to modules can be seen in section 8 of the [Module Guide](#). The tests that cover a specific requirement also cover the modules associated with that requirement.

## 11 Code Coverage Metrics

### 11.1 Front-end Coverage

The coverage results of the front-end unit testing can be seen in Figure [1](#). Perfect coverage was not achieved, but we believe our unit tests supplemented with our manual and usability tests provide sufficient coverage of the code.

| File                | % Stmts | % Branch | % Funcs | % Lines |
|---------------------|---------|----------|---------|---------|
| All files           | 63.83   | 36.86    | 67.34   | 63.91   |
| components          | 71.35   | 64.95    | 74.54   | 71.2    |
| DatasetInsights.tsx | 100     | 100      | 100     | 100     |
| FailurePopup.tsx    | 100     | 100      | 100     | 100     |
| Header.tsx          | 100     | 100      | 100     | 100     |
| LoadingSpinner.tsx  | 100     | 100      | 100     | 100     |
| LoginBox.tsx        | 95.23   | 70       | 80      | 95.23   |
| NewProjDialog.tsx   | 26.19   | 31.81    | 16.66   | 26.19   |
| ProgressData.tsx    | 100     | 100      | 100     | 100     |
| ProjectSection.tsx  | 100     | 50       | 100     | 100     |
| ProjectTitle.tsx    | 100     | 86.36    | 100     | 100     |
| QualityData.tsx     | 100     | 100      | 100     | 100     |
| RegisterBox.tsx     | 84.09   | 87.5     | 76.92   | 83.72   |
| SuccessPopup.tsx    | 100     | 100      | 100     | 100     |
| UserInfo.tsx        | 63.63   | 54.16    | 50      | 63.63   |
| WorkPerformance.tsx | 100     | 100      | 100     | 100     |
| components/ui       | 66.66   | 10.28    | 68.57   | 66.66   |
| avatar.tsx          | 100     | 100      | 100     | 100     |
| badge.tsx           | 100     | 100      | 100     | 100     |
| button.tsx          | 100     | 66.66    | 100     | 100     |
| card.tsx            | 88.88   | 100      | 66.66   | 88.88   |
| chart.tsx           | 33.33   | 7.84     | 41.66   | 33.33   |
| dialog.tsx          | 90.9    | 100      | 66.66   | 90.9    |
| input.tsx           | 100     | 100      | 100     | 100     |
| label.tsx           | 100     | 100      | 100     | 100     |
| progress.tsx        | 100     | 50       | 100     | 100     |
| radio-group.tsx     | 100     | 100      | 100     | 100     |
| textarea.tsx        | 100     | 100      | 100     | 100     |
| context             | 8.82    | 0        | 0       | 9.09    |
| AuthContext.tsx     | 8.82    | 0        | 0       | 9.09    |
| lib                 | 100     | 100      | 100     | 100     |
| utils.ts            | 100     | 100      | 100     | 100     |

Figure 1: Front-end Unit Testing Coverage Results

## References



## Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Reflection.

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?
2. What pain points did you experience during this deliverable, and how did you resolve them?
3. Which parts of this document stemmed from speaking to your client(s) or a proxy (e.g. your peers)? Which ones were not, and why?
4. In what ways was the Verification and Validation (VnV) Plan different from the activities that were actually conducted for VnV? If there were differences, what changes required the modification in the plan? Why did these changes occur? Would you be able to anticipate these changes in future projects? If there weren't any differences, how was your team able to clearly predict a feasible amount of effort and the right tasks needed to build the evidence that demonstrates the required quality? (It is expected that most teams will have had to deviate from their original VnV Plan.)