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

List of Tables

List of Figures

This document ...

3 Functional Requirements Evaluation

4 Nonfunctional Requirements Evaluation

4.1 Usability

4.2 Performance

4.3 etc.

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

9 Trace to Requirements

The traceability from tests to requirements can be seen in section 4.3 of the VnV Plan.

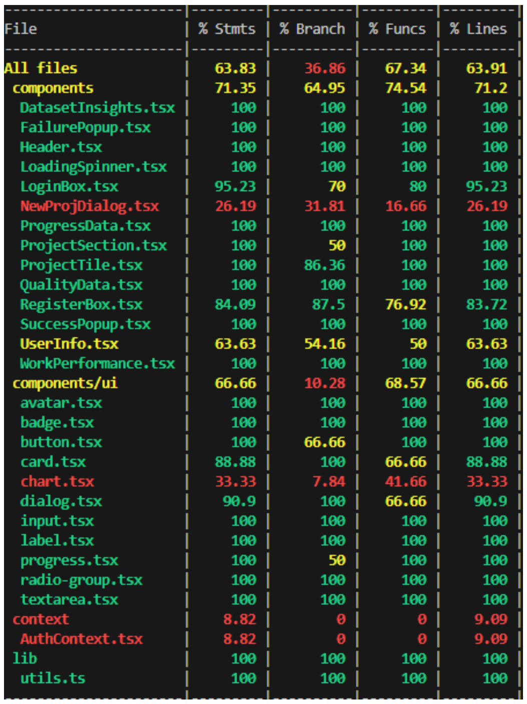
10 Trace to Modules

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 |
  ProjectTile.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.)