Reflection and Traceability Report on Software Engineering

Team #11, OKKM Insights
Mathew Petronilho
Oleg Glotov
Kyle McMaster
Kartik Chaudhari

1 Changes in Response to Feedback

1.1 SRS and Hazard Analysis

Table 1: Response to SRS & HA Feedback

Feedback/Concern	Source	Issue Link	Resolution/Action
POC should be aligned with the end goal of creating a satellite labeling platform	Supervisor	Issue #107	POC changed from where's waldo to satellite images of planes
Use previous accuracy results to identify experts	Supervisor	Issue #107	Plans changed to ensure an accuracy tracking system is implemented
Requirements should be prioritized	Peer	Issue #118	PR #350
Ambiguity in goals	Peer	Issue #119	No changes were made as we felt the specifics were covered in our requirements
Repeated problem statement, goals, and stakeholders	Peer	Issue #120	PR #308
Missing requirements traceability	Peer	Issue #121	PR #209
Add a context diagram	Peer	Issue #123	PR #225
Add intended audience section	Peer	Issue #133	PR #225
Undesired event handling needed	TA	Issue #213	PR #311
Likely changes should be provided	TA	Issue #122	PR #417
Document formatting issues	TA	Issue #212	PR #419

Continued on next page

Feedback/Concern	Source	Issue Link	Resolution/Action
Clarity about the new security requirements added	Peer	Issue #156	PR #309
More specific mitigation strategy for user input errors	Peer	Issue #153	PR #217
Include real world examples of similar failures to strengthen justification for mitigation strategies	Peer	Issue #162	We felt this was out of scope for the purpose of our documentation, so it was not implemented
Include strategy for handling risks associated with 3rd party libraries going out of date	Peer	Issue #173	Confirmed with the TA that we could assume stable 3rd party libraries would continue to be stable
Add meaning of terminology used in the document	Peer	Issue #174	PR #310
Add impact scores for each row in the FMEA	Peer	Issue #176	PR #218
Specify which hazards to address in which phase	Peer	Issue #157	PR #418
Formatting issues such as missing list of tables and table header	TA	Issue #230	PR #423

1.2 Design and Design Documentation

Table 2: Response to DD Feedback

Feedback/Concern	Source	Issue Link	Resolution/Action
Missing traceability between some FR and their modules	Peer	Issue #294	PR #315
Missing traceability between anticipated changed and modules	Peer	Issue #295	PR #314
Module descriptions should be more specific	Peer	Issue #301	We felt the specific were already covered in the MIS
Should be a flow of user interfaces	Peer	Issue #302	There was already a diagram included under the interfaces section within the module guide
State variables were missing for some modules	Peer	Issue #304	The way we approached it in our design was to not hold the information in state variables, rather we take in inputs through a form and process them
Include more details in the timeline	Peer	Issue #296	PR #428

Continued on next page

Feedback/Concern	Source	Issue Link	Resolution/Action
The assumption section was too ambiguous	Peer	Issue #297	No changes made as we did not feel they were ambiguous. We asked for an example but no response was given
Turn the use hierarchy table into a diagram for better understanding	Peer	Issue #298	PR #431
Specify additional security measures such as encryption techniques	Peer	Issue #299	PR #407
Add improvements to increase the accuracy of the data and reliability of labeling	TA	Issue #387	PR #405
Include class diagram for all modules	TA	Issue #388	PR #404
Formatting including broken links and fonts	TA	Issue #389	PR #413
Include how the weighted sum in getExpertise is calculated and how confidence is calculated	TA	Issue #390	PR #420
Include general exception handling	TA	Issue #391	PR #403

1.3 VnV Plan and Report

Table 3: Response to VnV Feedback

Feedback/Concern	Source	Issue Link	Resolution/Action
Add an objective related to security of the application	Peer	Issue #192	PR #225
All NFR being covered by a module is probably not a feasible check	Peer	Issue #201	PR #311
Include a reference to the hazard document when discussing the testing plan	Peer	Issue #202	PR #215
More specifics needed on contacting a QSA	Peer	Issue #203	PR #219
Clearer link of survey answers to success of usability tests	Peer	Issue #204	PR #220

Continued on next page

Feedback/Concern	**Source**	**Issue Link**	**Resolution/Action**
Not feasible to test all possible languages	Peer	Issue #210	Language related requirement was found to be out of scope, so the test was not conducted
Use preprocessing to improve the efficiency and bounding box accuracy	Supervisor	Issue #228	PR #38
Look into Meta's SAM and supplement with image meta data	Supervisor	Issue #228	Wanted to pursue this but ultimately did not have the time to do so
Split images based on concept density	Supervisor	Issue #228	PR #33
Formatting errors such as table headers, double references, typos	TA	Issue #243	PR #312
Add a nothing to label button to the front end for added user clarity	Professor	Issue #238	PR #10
Responsibilities of each members should be fleshed out more	TA	Issue #244	PR #425
Include how convergence will be tested	TA	Issue #245	PR #424
More detailed contextual popups and walkthrough, including a demo gif	Usability Tester	Issue #341	PR #27 PR #31
Clearer visual feedback and selection of label type	Usability Tester	Issue #341	PR #27
Clearer text for buttons to understand what they do	Usability Tester	Issue #341	PR #27
Bug where tool would get stuck if submit was clicked but no labels were made	Usability Tester	Issue #341	PR #27
Reselection of label type every time was annoying	Usability Tester	Issue #341	PR #27
Button to set zoom, contrast, brightness back to original	Usability Tester	Issue #341	PR #27
Remove white space	Usability Tester	Issue #341	PR #27
Remove help button upon project completion	Usability Tester	Issue #341	PR #27
Map unit test descriptions to their files	Peer	Issue #376	PR #414
Broken link to usability report	Peer	Issue #380	Updated action workflow
Add captions to tables	Peer	Issue #378	PR #406

Continued on next page

Feedback/Concern	Source	Issue Link	Resolution/Action
Include information about back-end unit tests	Peer	Issue #377	PR #432

2 Challenge Level and Extras

2.1 Challenge Level

The challenge level of our project is general. We were able to create a solution, but it took additional research and training.

2.2 Extras

- Usability Testing (view here): Conducted by allowing users to test the application interface and provide feedback to us through a questionnaire
- User Guide (view here): Create a guide demonstrating how to use the product and its various features

3 Design Iteration (LO11 (PrototypeIterate))

The front-end labeling tool began as an open-source tool with pre-built features. When we conducted usability testing, we did not get the results we were looking for in our survey. The specific feedback from the users can be seen in 1.3. The tool was very restrictive with the ways it could be modified and had unchangeable features that hurt usability. As a result, we decided to create an entirely new custom labeling tool from scratch. This allowed us to better address the feedback given to us, and also allows for flexible development in the future if we have any changes or additions we want to make. Specific changes included having clearer visual feedback, adding more steps to the tutorial, submitting labels bug fix, labels staying selected, condensed tool bar, and more detailed text for buttons.

For the back-end, several changes were made through the course of the project. Firstly, the initial consensus engine took 20 minutes to aggregate 200 labels. This was very slow and limited our ability to update confidence measures and efficiently serve images. To solve this, we decided to rewrite our consensus engine so that we could run it on a GPU. This resulted in a 95% reduction in energy usage and a new run time of just 8 seconds for 200 labels. Another back-end component that was modified was the storage of consensus data. We were initially storing m-by-n rows for each pixel of an m-by-n image. This resulted in slower retrieval and upload times and inefficient use of our database. Our solution was to pivot to a relational pixel-wise storage method, which reduced the rows needed to one row per image. Finally, our preprocessing algorithm started as a very simple method which split the submitted image into 300x300 pixel images. This algorithm resulted in problems with images that could not be split evenly, sometimes causing small strips of an image to be served to the user. We updated the algorithm to dynamically adjust split image size to be based on object complexity, so that each image served has similar complexity regardless of size. We also added an AI preprocessing step where we run the initial image through a model to get initial confidence levels for the objects we are looking for. The initial algorithm suffered from inaccuracy and redundant labels. We updated it to use convolution and pooling to reduce dimensionality. Sliding window detection combined with multi-scale analysis ensures accurate object detection at various sizes. Non-maximum suppression eliminates redundant bounding boxes, improving detection precision. Overlapping bounding boxes are removed based on intersection over union. Batch processing of the models output predictions was implemented to reduce the time it took to populate the database.

4 Design Decisions (LO12)

For our front-end, we decided to implement a responsive user interface that was designed for web browser first usage. This was a direct result of the constraint we outlined relating to our application being compatible with all modern web browsers. Access through a browser also facilitates easier user access, which is needed for a crowdsource based project. This decision made implementation easier as we did not need to worry about designing for other devices such as mobile.

The decision to switch to a custom-made labeling tool came from the limitations of the existing labeling tool that we were using. This change allows for easier modifications of the code in the future.

For our back-end, we chose to use a Bayesian interface-based consensus model to best satisfy the accuracy requirements and constraints we set. In scenarios requiring 90% confidence, our algorithm has higher accuracy and lower cost than alternatives, especially as the number of labeling tasks increase.

We chose to use a CNN model over other existing models because CNNs are specifically designed for image analysis. They excel at capturing spatial hierarchies and patterns using convolutional layers, making them ideal for object detection tasks. Other models like traditional ML algorithms (e.g. SVM, Random Forest) lack the ability to directly process image pixels and extract complex visual features. Additionally, CNNs outperform basic models in accuracy and efficiency for large-scale image datasets.

For database access from our backend, we chose to split database connectors into multiple classes so that different services would be able to reuse them. This led to less code duplication and better code maintainability.

Due to time constraints we needed to prioritize certain features. We needed to make a trade-off between the quality and quantity of features in our app. We decided to cut off some features that we were initially hoping to achieve, such as working payments and direct satellite image fetching. Implementing direct feedback on our existing features also took priority as stakeholders were expecting these to be addressed, leaving less time to implement other things.

5 Economic Considerations (LO23)

There is an existing market for our product, as well labeled datasets are at the core of quality AI models. As the AI industry continues to grow, plenty of people will be looking for datasets to train their models with. Also, manual labeling is slow, inconsistent, and high in cost. Our hybrid AI and crowd-sourcing pipeline significantly reduces time and cost to get a quality dataset.

Our product would require a heavy marketing campaign as without a substantial user-base our crowd-sourcing platform does not work. Our marketing campaign would be heavily focused on digital marketing through social media posts, forum posts, and online advertisements. There would also be a focus on foreign users, as the reward for labeling would be more substantial for some of them.

The current monthly cost of running our application is around \$15. This is without having our AI model running in the cloud. We project that a final version of the application could cost closer to \$100 per month considering higher traffic and model deployment.

The revenue model would be to charge a flat fee based on image size and complexity to clients who want their images labeled. We would take a 20% cut of this value, and the rest would be distributed to our labelers. They would be paid for every image they label. There is also the opportunity to introduce a subscription model for access to datasets we have already curated.

6 Reflection on Project Management (LO24)

6.1 How Does Your Project Management Compare to Your Development Plan

We adhered to exactly what was set out in our team meeting plan, having weekly meetings to discuss the progress everyone made and go over any outstanding deliverables. With regards to supervisor meetings, our lack of steady progress and a misaligned schedule meant we did not meet with our supervisor as much as we hoped to.

The team communication plan was also followed well. The team utilized Github issues to track progress and all concerns were communicated in our meetings or through our Teams chat.

Team roles were followed for the most part, but Oleg took on more responsibility with regard to the deployment of our application than Kartik.

Regarding our workflow plan, we stopped using the Github project board in the new year as we felt the normal issue tracking was enough. There was also some breaches in the approval process we laid out. There were instances where pull requests were merged without review due to time constraints.

With regards to our planned technology stack, we ended up using TypeScript instead of JavaScript on the front-end. Also, the database we chose to use was MySql. We never got the chance to make use of Meta's segment anything model, but there is still a possibility for that in the future.

6.2 What Went Well?

Our use of CI/CD to generate a pdf file for each of our latex files was very useful, and saved us a lot of time. Our consistent meetings were also helpful to track progress and get everyone's status. We used Github issue tracking regularly which contributed to a healthy work flow and clear responsibilities for each member. We had a very early start on the front-end, which allowed us to reach a stable interface fairly early. This meant that we could focus on changes and tweaks recommended through user feedback. From a technology perspective, integrating our front-end with Vercel was very effective. It allowed us to see what our staged branches would look like in production, and also told us if anything needed to be addressed before pushing to production.

6.3 What Went Wrong?

One of the main problems with our project was that we overestimated what we would actually be able to do. This led to a lot of unnecessary documentation and time spent fleshing out ideas that were never implemented. Another problem we encountered was when our production database was wiped and we did not have any backups for the time period we needed. This resulted in extra time spent redefining tables and populating data. A huge problem in the development of the application and the documentation was the deadline heavy workload. A lot of team contributions were done at the deadline, making it really hard to give any feedback or make iterative improvements.

6.4 What Would you Do Differently Next Time?

For the next project, we would look to tighten our scope to just a few key features, rather than list all the features our application could include. Starting too ambitiously led to cutting features due to time constraints, affecting project quality. With regards to data storage, we would ensure that a development database is set up and used in our testing environment so that we avoid wiping out our production data. We can also regularly store backups of the database in case we ever need to revert. Finally, we would look to have more group working sessions. Not only would this force team members to start on their work before the day of the deadline, but it allows for better collaboration as ideas can be shared in real time. We held a meeting like this toward the end of the project and it was very effective.

7 Reflection on Capstone

7.1 Which Courses Were Relevant

- SFWRENG 2AA4: This course gave us a basis in software development, software specification, and testing software.
- SFWRENG 3RA3: This course helped with creating quality requirements for our SRS.
- SFWRENG 3A04: This course helped with creating our design documents and choosing the architecture for our project.
- SFWRENG 3BB4: This course helped with implementing parallel processes in our back-end.
- SFWRENG 4HC3: This course helped us to maximize the usability of our front-end.
- COMPSCI 4ML3: This course helped us create our CNN and consensus models.
- SFWRENG 3DB3: This course helped us to create our relational database design and make use of SQL queries.
- COMPSCI 4SD3, STATS 2D03, STATS 3D03: These courses helped us create our consensus algorithm.

7.2 Knowledge/Skills Outside of Courses

- Front-end Development: We needed knowledge on how to create the front-end web application using TypeScript and React.
- Github CI/CD tools: We needed to do extra research to figure out how to set up automatic file generation and regression testing with Github.
- API design: Knowledge about APIs was needed to ensure there was seamless communication between each component of our project.

- GPU Programming: Additional resources were used to understand how we could implement our consensus algorithm to use a GPU.
- Statistics: Understanding of more complex statistics concepts were needed to design our consensus algorithm. This included consulting scientific papers to aid in our implementation.