Project Title: System Verification and Validation Plan for Natural Language Processing for Mental Health Risk Prediction

Team 13, The Cognitive Care Crew
Jessica Dawson
Michael Breau
Matthew Curtis
Benjamin Chinnery
Yaruo Tian

November 3, 2023

Revision History

Date	Version	Notes
11/03/2023	1.0	Revision 0

Contents

List of Tables

1 Symbols, Abbreviations, and Acronyms

symbol	description
UQAM	Université du Québec à Montréal
RMSE	Root Mean Square Error
MZOE	Mean Zero-One Error
MAE	Mean Absolute Error

2 General Information

2.1 Summary

The CLEF eRisk competition, better known as the Early Risk Prediction on the internet competition, is an organization that hosts a yearly event where numerous research teams come together to explore new strategies and applications of early detection technologies, particularly in the field of health and safety. The team at McMaster has partnered alongside the University of Quebec in Montreal to help leverage their existing research and work done in prior years of the eRisk competition to help design a new system entry for this year. Our team will not be in charge of the system operations portion of the project, but instead will be focused on designing new strategies and implementations for the Natural Language Processing portion of the project. The NLP will be responsible for analyzing textual input from real user data, and will use its provided training data and algorithms to determine the probability of the user showing signs of a selected mental health issue. The competition normally consists of three different tasks, all of which require different Natural Language processing techniques, to help us form our strategies, we are using data from the prior year competition, which was used to find symptoms for depression, pathological gambling and eating disorders.

2.2 Objectives

The primary objectives for our current process is to build confidence regarding the various techniques and libraries required to efficiently analyze the datasets provided by the eRisk Competition. The testing outlined in this document is help demonstrate our familiarity and competency to adapt to the various challenges presented by eRisk, whether that be through sentence ranking, early detection through post history, or estimation of severity level through user submissions. This is meant to build confidence for our stakeholders on our code efficiency, accuracy and reliability. The exact number of challenges that the team will participate in is variable, as the scope ensures that we will compete in at least one challenge, as even though we now have a larger team than prior years, past teams have not always been able to take on all three challenges, extending ourselves to take on all tasks would be going beyond SRS expectations. An additional objective is to increase efficiency and accuracy of the existing work done by the team in prior years, this is

a goal for both the team and our stakeholders at UQAM, but it is not a formalized metric in the SRS, as there is no defined expectation of how much better our project is supposed to preform.

2.3 Relevant Documentation

- Problem Statement

The Problem statement document abstractly identifies the problem to be solved, characterizing it in terms of its inputs and outputs, which gives context to the related environment and stakeholders.

- Development Plan

The Development Plan contains the project development overview including team roles, team meeting and communication plans, git workflow and project schedule.

- SRS

The Software Requirements Specification identifies the various functional and non-functional requirements related to the project, while providing context for the benefits and usage scenarios of the project.

- Hazard Analysis

The Hazard Analysis identifies various hazards and obstacles to the project's development and operation, along with mitigation strategies to help deal with those potential hazards. It carries a focus on ensuring the project can be delivered to the standards of the eRisk competition while maintaining safety and privacy of user data.

- Module Guide

The Module Guide decomposes the eRisk NLP project into numerous modules based on the principle of information hiding and separating data structures, in order to help increase understandability of various project components.

- Module Interface Specification

The Module Interface Specification identifies modules found in the Module Guide and decomposes them into more primitive data types and functions, in order to better observe the module functionality and ensure it's behaving as intended.

- McMaster AI Guidelines

McMaster's Provisional Guidelines on the use of AI provides useful rulesets on generative AI that helps our team ensure the safe and responsible use of AI practises.

- eRisk 2022 Paper

The prior efforts done by the UQAM team in the eRisk competition were instrumental in our team's foundation of understanding and served as a jumping off point for our efforts.

3 Plan

This section will outline the plan for verification and validation of a number of different components within this project (3.1). This section will first give an overview of the members that are considered part of the verification and validation team and what their roles and involvement will consist of. Next there will be subsections regarding verification of the project's SRS (3.2) along with the design plan (3.3) and the verification and validation plan itself (3.4). This section will then outline the verification plan that has been created for the implementation of the project and model (3.5). Automated Testing and verification tools that will be used throughout this project is then the next subsection that is outlined in 3.6 followed lastly by the validation plan for our software (3.7).

3.1 Verification and Validation Team

The verification and validation team will consist of our core team members (Matthew, Jessica, Ben, Yaruo and Michael) along with our professor, our TA, Marie-Jean and Diego from our Montreal team, and Professor Mosser. Our core team will be responsible for creating test suites that ensure correctness in our solution and that will catch possible bugs and issues that

may arise. The team will be responsible for creating suitable edge cases to evaluate the correctness of our work along with general automated test suites that will be automatically deployed when new code is pulled.

The core team will be responsible for creating all test suites, along with executing them and documenting the results. We will also be responsible for making any changes that are required after testing our code. All core team members will have a hand in all sections of testing but different team members will have different focused responsibilities. Firstly, all core team members will be responsible for documenting the results of the automated test suites when their code enters the repository through a pull request. More specifically, Yuiro and Micheals main responsibility will be creating a set of tests including edge cases for our NLP model that will ensure that our model functions as expected for a wide variety of input data. They will be required to create test suites along with automated test suites that will be run periodically when pull requests happen. Jessica, Ben and Matthew on the other hand will have the primary responsibility for training the data vs a training data set in order to determine the accuracy of the model. They will also have the responsibility to ensure code structure in the test suites that are created along with organizing the suites while Yaruo and Michaels main role is coming up with and creating the tests.

The team will meet with Professor Mosser and their TA as well periodically throughout each milestone in order to discuss the requirements for our project regarding the current milestone and to help solidify what is expected of us. This is an opportunity for us to ask any questions we may have as well. We also will meet with Marie-Jean and Diego from the Montreal team periodically to help guide us with regard to requirements of our project and the eRisk competition along with what validation means to them. They will help guide us to what the important things are to focus on within our project. Team 8 will also provide us feedback for every milestone. Lastly, the actual competition itself will be our final validation step when it tests our model against a new set of data and reports how our model performs.

3.2 SRS Verification Plan

All of the members of our core team will be taking part in the SRS Verification Plan along with Group 8, Professor Mosser and our TA. We will verify the contents of our SRS by comparing the requirements outlined by the eRisk competition along with our team in Montreal with the requirements stated in our SRS document. This will ensure that we have covered everything from our SRS document and we can see if there is anything new we must add. Most of our SRS verification plan will involve inspections from us but also from the team in Montreal along with peer reviews and ad hoc feedback from group 8.

Since our project is unique in the fact that we are submitting to a competition with a rigid output and guideline structure, we will also have to compare the rubric for the SRS document with what we have done in our original SRS document to ensure that we have covered all the required checkpoints. On top of this, we will verify our SRS document by going over the feedback given to us on our SRS revision 0 from our TA along with the feedback we received from group 8 regarding our SRS revision 0. Lastly, when we are verifying our SRS document we will talk with Professor Mosser and ask him any questions that arise during the reviewing and verification process within the group.

These are some of the major areas we would like to cover in our SRS verification plan to ensure we have a test for each of these components:

- Is a functional overview of the system provided in the SRS?
- Are high-level usage scenarios included?
- Have we specified the software environment and all its elements?
- Are the limitations and exclusions accurate and all encompassing of the true limitations and exclusions of our project?
- Are the inputs and outputs of the system specified?
- Is there any unnecessary overlap in our requirements?

- Are there any other components outside of Text Pre-Processing, Vectorization, Prediction and Output? For those 4 sections, are there any additional functional or non-functional requirements that are missing?
- For every function that is outlined, are the inputs specified sufficient enough to perform the required functionality?
- Do any requirements conflict with each other or does each requirement avoid conflicts with other requirements?
- Are all detailed usage scenarios covered in our document and are completed in adequate detail or are there others we could add?
- Does each component and requirement have a priority?
- Are there any chances in our prioritization chart we should add due to changes in our code since the creation of our original SRS (revision 0)?
- Are the requirements clear enough that it would be possible to give to an independent team for implementation and they would be able to understand the requirements?
- Is every requirement testable?
- Is the verification and acceptance criteria specific enough? If not, how can you add quantifiable metrics to each point?
- Have we defined every definition, abbreviation and acronym in our Glossary?
- Are all imposed technical choices listed in the document?
- Are the risks that may be present in this project along with their mitigation analysis included in this document?
- Is the "Requirements Process and Reports" accurate to what is expected by each of the teams listed in the "Requirements Process and Reports" section?
- Does the Context and Overall Objectives section paint an accurate enough description of the project's context and objectives?

3.3 Design Verification Plan

Our plan for our design verification will be to go through our SRS document and make sure that each of our outputs and inputs that we planned on having are accounted for and included in our design planning and document. We will also look at reviews given to us by Group 8 on our SRS document to consider any changes we may want to add or things to consider for our design. We will go through the same process for feedback received from our TA on our SRS report. We will also utilize the guidance of Marie-Jean and Diego in order to verify that we are on the right path with our design. They can help guide us in our design and verify that it checks all the boxes and functionalities that are required for the competition.

Below I will go over some design verification questions that will be asked regarding the functions of each of the major components in our design.

Text Pre-Processing Component:

- Does the system tokenize text by dividing it into individual units of words and or sub-words?
- Does the system convert generated tokens into lowercase to preserve consistency?
- Does the system remove stop word tokens (common words that do not commonly have an effect on the meaning)?
- Does the system reduce tokens to their root forms (Ex: “moving” to “move”)?
- Is the output from this component usable by the Vectorization Component?

Vectorization Component:

- Does the system convert tokens into numerical vectors in this stage?
- Are these numerical vectors usable by the Prediction Component of the model?

Prediction Component:

- Is there a created training model within the design?
- What percentage of the given data is used for training vs testing?
- Is there a prediction algorithm that is implemented into the solution? What is the level of accuracy and what is acceptable levels of accuracy (this will be determined further down the road with our team in Montreal on what we deem acceptable)?
- Does the system make predictions using the vectorization tokens, the training model, as well as with the created prediction algorithm?

Output Component:

- Does the system output the predictions to the console?
- Is the output produced understandable by the project team?
- Is the output in a format that a healthcare worker would be able to read and interpret the results?

3.4 Verification and Validation Plan Verification Plan

For our Verification and Validation Plan Verification Plan we will be making use of the reviews given to us for our VnV Plan from group 8 as well as from our TA. This will help us gain some insight on areas that may be missing or need more work within the document. We will also be going through our SRS document and ensuring we have created at least 1 test for each functional and nonfunctional requirement. It is important to ensure that we have sufficient tests for our requirements. We would also like to do some mutation testing in our code to determine what changes will affect the tests we have created and in what way. It will also show us that there is a possibility we are missing some tests that should be included if a mutation that we would expect to result in a change or failure, results in no failed tests.

Below is a checklist of the type of questions and topics our team would like to cover as part of our Verification and Validation Plan Verification Plan.

- Is the brief summary of the software in the summary section enough to get an understanding of the software that is being tested and its general functions?
- Are the objectives for the VnV plan clearly outlined and is each objective covered at some point throughout the document?
- Is the planning section detailed with checklists for each section and steps that will be taken for each plan?
- Is there at least one test created for each requirement in the SRS document?
- Are all functions of each component being tested? This includes the Text Pre-Processing, Vectorization, Prediction and Output Components.
- Are there a wide variety of tests including edge cases? Do the non-functional requirements contain tangible quantifiable values?
- Is the traceability between test cases and requirements clearly shown in section 4.3?

3.5 Implementation Verification Plan

The large majority of the implementation for this project will revolve around achieving an acceptable level of accuracy in our system's predictions. This document goes more in depth about how we will do this in our implementation in section 4.1.1 but the basis of this process to ensure acceptable model accuracy levels is done through train/test splits, training models on the test sets, predicting the values of test data with these models, and then comparing these predicted values with the known expected values. All of the sections just listed are key areas of the implementation and will need to be tested in order to verify our implementation. There are 4 tests that will be run, typically in tandem with each other in one automatic test suite as outlined in 4.1.1. The largest area will be involving training and testing the data. Data will be provided by eRisk and the data will be used 3 times using different splits for each copy regarding what data is used for training vs testing. These predictions on the testing data will then be evaluated on a relevant set of metrics in order to test the program's accuracy and implementation. This

set of metrics has yet to be determined due to the fact that the eRisk tasks have not been released yet for this year so we are unclear what metrics would be appropriate to evaluate our model at the current time. We will also be running a series of unit tests on our program in order to test the implementation of each part and function of the code. These tests will be recorded in the unit testing plan.

As new code is added to our repository and as we work on our model, we will be using implementation verification techniques like code walk-throughs and code inspections periodically. Every time a new pull request is made, a fellow teammate will be required to perform a code inspection for that pull request and give feedback. If there is something unclear, the two teammates will meet and the reviewer will receive a code walk-through from the reviewee. We will also perform code walk-throughs for new changes made to the model when we meet as a team. Lastly we plan to have continuous integration that will allow us to run automated testing for every pull request that gets created as well as utilizing Fake9 as mentioned below to ensure our implementation follows proper coding standards.

3.6 Automated Testing and Verification Tools

There will be three main tools the team will be using for automated testing and verification. Firstly, we will be using Flake9 which is a fork of Flake8. This will help us in following proper coding standards and prevent problems arising regarding topics such as syntax errors, typos, incorrect styling, bad formatting and more. This will also help save time for when we do peer code reviews as a team. We will also be creating test suites using Pytest since it is a free open-sourced, simple and scalable Python-based Test Automation Framework. We will be using GitHub Actions in order to implement continuous integration and continuous delivery into our project. This will allow us to automatically build, test, and deploy our pipeline. This will create a pipeline that allows us to build and test every pull request when they happen and also when we deploy and merge a pull request onto another branch.

3.7 Software Validation Plan

There are a multitude of methods and parts to our software validation plan. Firstly, our primary validation method will be the actual process of com-

peting in the eRisk competition. During the competition they will take our model and test it with new data and observe the output. This will be the final and largest validation step for us that will help validate the correctness and precision of our design. We will also have a set of training data that we will use throughout the design process to train against that we can use to test how close our results are to the expected results, validating how accurate our design is.

Another large aspect of the software validation plan will be our review sessions and meeting with Marie-Jean and Diego throughout the project. This will be a time for them to look at our software and run it themselves. A large checkpoint when they will have an opportunity to have our project demoed for them is around the Rev 0 demo. This will give them an opportunity to validate that our project passes all the requirements of the competition or if there is something else we need to add or change. This is an important time for us to take this feedback and use it in order to improve our project. This will also be a time for us to ask questions and for them to confirm the requirements are all being met in our design. These questions and requirements would largely surround the overall output of the design and the overall process (largely focusing on the training of our data and our prediction model we create and use). It would also include validation of each of the program's components individually and as a collective (components being the Text Pre-Processing, Vectorization, Prediction and Output components).

Before that however, we will meet with professor Mosser to discuss if he thinks that all the requirements documented line up with and are accurate to the requirements outlined for the course. We will do this with Marie-Jean and Diego as well with respect to the requirements from the competition. We will also go over the SRS document one last time to ensure that every requirement is met.

Regarding our demo with Professor Mosser, Marie-Jean and Diego we will include the following questions:

- Is the output format what is expected by the competition?
- Are there any requirements not being met within the competition (question for Marie-Jean and Diego)?

- Are there any requirements that are not being met regarding the project rubric from McMaster (question for Professor Mosser)?
- Is our text pre-processing component sufficient in which we use tokenization, lowercasing, stopword removal, and stemming or Lemmatization? If not, what should we add to our text pre-processing component or what methods that we currently have employed are not needed?
- Is our current method of vectorization appropriate for the task we are completing in your opinion? If not, what would you change or what different approaches do you recommend we look into.
- Are there any areas of our code or processes we take that are not allowed in the competition or do not follow the competitions standards?
- What are your thoughts on our topic modeling approach? Do you have any suggestions?
- Are the metrics we use for determining accuracy appropriate in your opinion? If not, what form of metrics should we be using? What level of accuracy would you consider acceptable for this metric?
- In your opinion is the run time of our program to output the results acceptable? If not, what would be an acceptable time?

4 System Test Description

4.1 Tests for Functional Requirements

Our main requirements for the system are built around achieving an acceptable level of accuracy in our system's predictions. This is the primary focus of this section.

4.1.1 Ensuring Model Acceptability

The system is based in machine learning and will therefore need to be tested and evaluated using a typical machine learning evaluation method: creating train/test splits, training models on the test sets, predicting the values of test data with these models, and then comparing these predicted values with the known expected values. This section first includes information explaining

how the team plans to go about this process as well as identifying areas of uncertainty before providing relevant tests.

4.1.1.1 Training and Test Data

Training data given by eRisk will be split 80/20 into train and test sets. Each task has some concept of higher or lower risk, either in the form of positive/negatives, survey answers, a continuous scale, or something else. Depending on the task a metric will be created from expected values representing this risk (an example may be averaging survey answers) it is unclear what these metrics will involve currently as eRisk has not yet released this year's tasks. Train and test splits will be created to have similar distributions of this risk metric (ie. similar numbers of low/mid/high risk individuals). How similar the distributions should be is as of yet unknown and will require further experimentation when eRisk releases their datasets for this year.

When testing the system three different train/test splits will be produced, all splits will be formed on the full training data provided by eRisk, thus giving three copies of said training data with the train and test subsets being different for each copy. These copies will try to maximize the difference between their splits. The model will be trained on a training split and then will give predictions on the test set. These predictions will be evaluated on a relevant set of metrics (outlined below in section 4.1.1.2), and an average of the metrics across all three splits will be used to determine model performance.

4.1.1.2 Metrics

As the eRisk tasks have not yet been released it is unclear exactly what form the expected and predicted output will take, so it is unclear what metrics will be used to evaluate our models. If a task is a repeat from past years, metrics similar to the ones used in previous versions of the task may be used. Here are some metrics that have been deemed potentially useful to employ, whether they will be used depends on the nature of this year's eRisk tasks. Other measures will likely be added upon the release of the eRisk tasks:

1. MAE (Mean Absolute Error) was used by the UQAM team in a 2022 task about eating disorders (Saravani et al., 2022). MAE measures the average distance between a predicted value p_i in predictions P and a ground truth (expected value) t_i in ground truths T for n data points:

$$\text{MAE} = \sum_{i=1}^{n} \frac{|t_i - p_i|}{n}$$

2. MZOE (Mean Zero-One Error) was also used in the UQAM's 2022 eating disorder submission (Saravani et al., 2022). MZOE measures the proportion of incorrect predictions:

$$\text{MZOE} = \frac{|\{p_i \in P : t_i \neq p_i\}|}{n}$$

3. F-Score was used in the UQAM team's 2021 submission for the detection of problem gambling task (Maupomé et al., 2021). F-Score is a combination of two other measures, precision and recall, and is used to measure the accuracy of positive/negative tests while taking into account false positive and negatives:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

$$\text{F-score} = 2\frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

4. RMSE (Root Mean Square Error) is a common measure of the deviation between predicted values and expected values, it somewhat analogous to MAE:

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(t_i - p_i)^2}{n}}$$

5. R^2 measures how well the expected values are predicted by the model by comparing how much variation the predicted values capture against the variation in the expected values:

$$SS_{res} = \sum_{i=1}^{n} (t_i - p_i)^2$$

$$SS_{tot} = \sum_{i=1}^{n} (t_i - \text{mean}(T))^2$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

4.1.1.3 Acceptable Error Metric Values

The bounds for which error measures are considered acceptable will be based on the results other teams achieved in past eRisk competitions if the given task is a repeat. Our model should perform better than previous years. If the task is new, bounds will be determined through research into the accuracy of clinicians in these areas and the involvement of expert knowledge in the form of psychology faculty from the Université du Québec à Montréal. Our model should represent an improvement on clinicians' diagnostic capabilities.

4.1.1.4 Tests

In practice these four tests will rarely be run separately but as one automatic test suite as the output of each test is fed to the input of the next.

1. FRT-ACC1-1

   Control: Automatic

   Initial State: No models have been trained by the system yet, the eRisk training data is available to the system.

   Input: The eRisk training data.

   Output: Three train/test splits where each train/test duo is composed of disjoint train and test sets whose union is all of the data in the original eRisk training data.

   Test Case Derivation: The system should properly split the data into train and test sets.

   How test will be performed: The eRisk training data will be fed into the system, the system will create the splits. For each split a script will check that no element exists in both the train and test split and that all training data is accounted for between both sets.

2. FRT-ACC1-2

   Control: Automatic

   Initial State: No models have been trained by the system yet, three train/test splits have been created.

   Input: The three sets of training data.

   Output: The parameters and/or objects (ie. class instances) that represent trained models.

   Test Case Derivation: The system should train models on the train splits.

   How test will be performed: The three training sets will be fed into the system, the system will train a model on each set. A script will check that the models exist and can produce predictions for data in the training sets.

3. FRT-ACC1-3

   control: Automatic

   Initial State: Three versions of the model have been trained on the relevant train sets.

   Input: The three corresponding test sets into the corresponding models.

   Output: Three sets of predictions.

   Test Case Derivation: The system should produce some sort of prediction.

   How test will be performed: The test data will be fed into the system and the system will use its trained models to produce a set of predictions. A script will check that there is a prediction for each test data point.

4. FRT-ACC1-4

   Control: Automatic

   Initial State: Three sets of predictions have been produced by the system.

Input: The three sets of predictions and corresponding expected values.

Output: A set of metrics formed by averaging together the metrics for each of the three sets, the specific metrics used will depend on the nature of the specific eRisk task (see section 4.1.1.2).

Acceptability of Output: For the system to be acceptable to stakeholders these metrics must be better than some baseline value determined from either past eRisk submissions for similar tasks or through research into typical clinical results (see section 4.1.1.3).

Test Case Derivation: The system must reach specific accuracy measures to be considered acceptable to stakeholders.

How test will be performed: Predictions and expected values will be given to system, which will calculate the relevant error metrics and how much they outperform baseline error metrics.

4.2 Tests for Nonfunctional Requirements

4.2.1 Usability Requirements

1. NFRT-U-1

   Type: Dynamic, Manual

   Initial State: System is completed and trained

   Input/Condition: The system will be ran on a Windows desktop/laptop and macOS desktop/laptop device with the correct environment setup

   Output/Result: The system should generate an expected result

   How test will be performed: After setting up the environment on to both macOS and Windows machines, the system will be ran on both environment with the same command. Test will pass as long as an expected result is generated from both machines.

4.2.2 Safety and Security Requirements

1. NFRT-SS-1

   Type: Dynamic, Manual

Initial State: The system is completed and trained

Input/Condition: The system will be ran on the developer's computer

Output/Result: The generated result should not overuse the processing power of the developer's computer

How test will be performed: CPU, GPU, RAM and live power usage will be monitored while running the software. Usage of all aspects should not increase more than 20 percent.

1. NFRT-SS-2

   Type: Automatic, Dynamic

   Initial State: The system has been trained and is awaiting data to form predictions on.

   Input/Condition: A set of data that the system can predict on.

   Output/Result: The resulting predictions, in a form where no sensitive data from the input is present in the output.

   Test Case Derivation: Sensitive in this case refers to the post history of the subjects eRisk provides as training and test data. If these posts can be reconstructed from any part of our system outputs it is possible the identity of the individual could be discovered. This represents an unacceptable breach of privacy and can not happen.

   How test will be performed: After the system has produced it's predictions a script will be run that takes all posts in the input data, forms a string out of all consecutive three word triplets (ie. "hello my good friend" forms "hello my good" and "my good friend"), both with and without processing (removal of stopwords, punctuation, etc.), and scans the output to ensure that none of these triples are present.

4.2.3 Legal Requirements

1. NFRT-L-1

   Type: Static, Manual

   Initial State: The source code and documentation is prepared

Input/Condition: The user reviews the entirety of the source code and related documentation

Output/Result: Copyright licenses, appropriate credits and/or MIT license must attached

How test will be performed: The testers will review the entirely of the project and check to see if appropriate copyright licenses and/or credits are included for resources that require them

4.3 Traceability Between Test Cases and Requirements

Requirement traceability from S2 of SRS to testing in this document.

Table 1: Traceability Between Functional Test Cases and Functional Requirements, FR-1 to FR-10

Test IDs	**Functional Requirement IDs**									
	FR1	**FR2**	**FR3**	**FR4**	**FR5**	**FR6**	**FR7**	**FR8**	**FR9**	**FR10**
FRT-ACC1-1						X				
FRT-ACC1-2							X			
FRT-ACC1-3	X	X	X	X	X			X		
FRT-ACC1-4									X	X

Table 2: Traceability Between Non-Functional Test Cases and Non-Functional Requirements

Test IDs	**Non-Functional Requirement IDs**			
	GR1	**GR2**	**SR1**	**SR2**
NFRT-U-1	X			
NFRT-SS-1			X	
NFRT-SS-2				X
NFRT-L-1		X		

5 Unit Test Description

[This section should not be filled in until after the MIS (detailed design document) has been completed. —SS]

[Reference your MIS (detailed design document) and explain your overall philosophy for test case selection. —SS]

[To save space and time, it may be an option to provide less detail in this section. For the unit tests you can potentially layout your testing strategy here. That is, you can explain how tests will be selected for each module. For instance, your test building approach could be test cases for each access program, including one test for normal behaviour and as many tests as needed for edge cases. Rather than create the details of the input and output here, you could point to the unit testing code. For this to work, you code needs to be well-documented, with meaningful names for all of the tests. —SS]

5.1 Unit Testing Scope

[What modules are outside of the scope. If there are modules that are developed by someone else, then you would say here if you aren't planning on verifying them. There may also be modules that are part of your software, but have a lower priority for verification than others. If this is the case, explain your rationale for the ranking of module importance. —SS]

5.2 Tests for Functional Requirements

[Most of the verification will be through automated unit testing. If appropriate specific modules can be verified by a non-testing based technique. That can also be documented in this section. —SS]

5.2.1 Module 1

[Include a blurb here to explain why the subsections below cover the module. References to the MIS would be good. You will want tests from a black box perspective and from a white box perspective. Explain to the reader how the tests were selected. —SS]

1. test-id1

Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

Initial State:

Input:

Output: [The expected result for the given inputs —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

2. test-id2

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

3. ...

5.2.2 Module 2

...

5.3 Tests for Nonfunctional Requirements

[If there is a module that needs to be independently assessed for performance, those test cases can go here. In some projects, planning for nonfunctional tests of units will not be that relevant. —SS]

[These tests may involve collecting performance data from previously mentioned functional tests. —SS]

5.3.1 Module ?

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

   Type: Automatic, Dynamic

   Initial State: The system has been trained and is awaiting data to form predictions on.

   Input/Condition: A set of data that the system can predict on.

   Output/Result: The resulting predictions, in a form where no sensitive data from the input is present in the output.

   Test Case Derivation: Sensitive in this case refers to the post history of the subjects eRisk provides as training and test data. If these posts can be reconstructed from any part of our system outputs it is possible the identity of the individual could be discovered. This represents an unacceptable breach of privacy and can not happen.

   How test will be performed: After the system has produced it's predictions a script will be run that takes all posts in the input data, forms a string out of all consecutive three word triplets (ie. "hello my good friend" forms "hello my good" and "my good friend"), both with

and without processing (removal of stopwords, punctuation, etc.), and scans the output to ensure that none of these triples are present.

5.3.2 Module ?

...

5.4 Traceability Between Test Cases and Modules

[Provide evidence that all of the modules have been considered. —SS]

References

Diego Maupomé, Maxime D. Armstrong, Fanny Rancourt, Thomas Soulas, and Marie-Jean Meurs. Early detection of signs of pathological gambling, self-harm and depression through topic extraction and neural networks. In *Conference and Labs of the Evaluation Forum*, 2021. URL https://api.semanticscholar.org/CorpusID:237298571.

Seyed Habib Hosseini Saravani, Lancelot Normand, Diego Maupomé, Fanny Rancourt, Thomas Soulas, Sara Besharati, Anaelle Normand, Sébastien Mosser, and Marie-Jean Meurs. Measuring the severity of the signs of eating disorders using similarity-based models. In *Conference and Labs of the Evaluation Forum*, 2022. URL https://api.semanticscholar.org/CorpusID:251471885.

6 Appendix

6.1 Symbolic Parameters

None.

6.2 Reflection

1. What knowledge and skills will the team collectively need to acquire to successfully complete the verification and validation of your project? Examples of possible knowledge and skills include dynamic testing knowledge, static testing knowledge, specific tool usage etc. You should look to identify at least one item for each team member.

   The following knowledge and skills will need to be acquired by the team to sucesfully complete the verification and validation for the project.

   1. Github Actions
   2. Pytest
   3. Mental Health Diagonostic Basics (For creating expected values)
   4. Python script for parsing and checking files
   5. System performance diagonistic skills

2. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?

   1. Michael will acquire the skill/knowledge required for this skill due to being the owner of the repository. Approaches for this skill/knowledge can be to either research the topic on the github actions webpage or watch YouTube tutorials to find the required skillset.

   2. Jessica will acquire the skill/knowledge required for this skill due to being interested in the skill. Approaches for this skill/knowledge can be to either research the topic on the pytest documention webpage or watch YouTube tutorials to find the required skillset.

   3. Ben will acquire the skill/knowledge required for this skill due to being interested in the skill. Approaches for this skill/knowledge can

be to either research the topic with various papers or watch YouTube lectures to learn more about the required skillset.

4. Matthew will acquire the skill/knowledge required for this skill due to being interested in the skill. Approaches for this skill/knowledge can be to either research the skill using the documentation of some python libraries for parsing documents or watch YouTube tutorials to find the required skillset.

5. Yaruo will acquire the skill/knowledge required for this skill due to being interested in the skill. Approaches for this skill/knowledge can be to either research the topic on various papers or watch YouTube tutorials to find the required skillset.