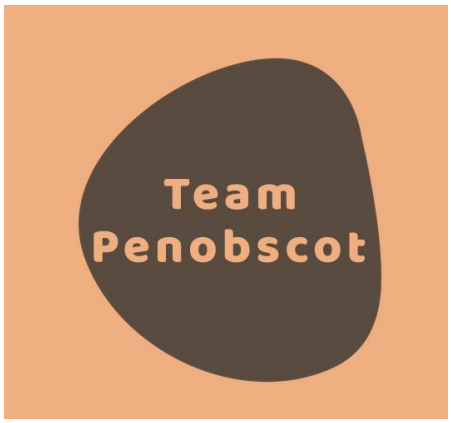


Seismic-radar toolbox GUI

System Requirements Specification

Dr. Chris Gerbi

Team Penobscot
Adam Farrington, Alex Thacker, Jens Hansen, Nathan Gazey
October 23, 2019

Table of Contents

1. Introduction

1.1 Purpose of This Document

This is the formal agreement between Dr. Gerbi and Team Penobscot that describes in detail the product to be delivered. Dr. Gerbi has requested assistance in designing and building the front end of a program that he has been working on with his colleagues. Included in this document are the description, scope, and agreed upon requirements of the desired software. This document also describes the timeline on which Team Penobscot will produce the desired software. This timeline will describe all documents that will be generated and software artifacts that will be produced.

1.2 References

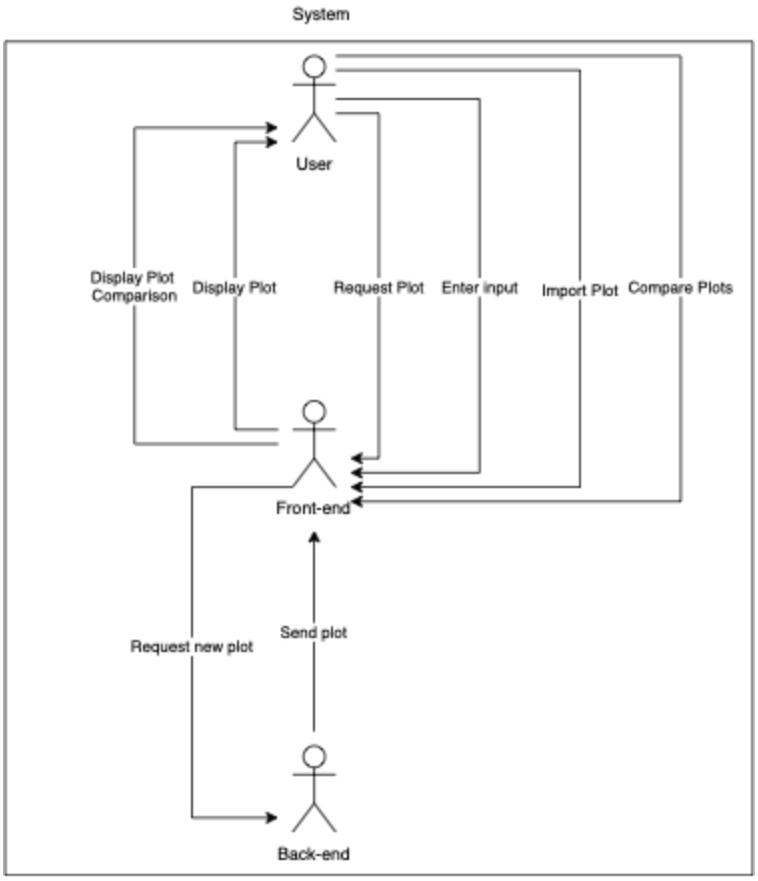
1. A slideshow describing the purpose of the backend software.
2. Software which generates cross platform executables out of python code.

1.3 Purpose of the Product

Dr. Christopher Gerbi is a professor for the School of Earth and Climate Sciences at the University of Maine. Dr. Gerbi, along with Dr. Seth Campbell, Ann Hell, and Steven Bernsen, has been developing backend software that is able to predict radar and seismic signals from simulated surveys sites. This software is nearing completion and currently is operated through the use of command line inputs. This is not an ideal interface for the software. Thus, Dr. Gerbi has partnered with the 2019 computer science capstone course at the University of Maine to create a Graphic User Interface (GUI) for new software. This GUI should be able to simplify the use of the program and allow for widespread distribution to the academic community.

1.4 Product Scope

The system being produced will act as a direct link from the existing backend software and the user. The diagram below shows the primary connections that will exist between each of the actors (being the user, backend and frontend).



2. Functional Requirements

This section of the document outlines all functional requirements derived from the user stories generated jointly by the customer (Dr. Gerbi) and development team (Team Penobscot). Each use case will map to one or more test cases that will mark completion of that requirement.

2.1 Use Cases

Number	1	
Name	Import Plot	
Summary	The User imports a plot into the GUI.	
Priority	3	
Preconditions		
Postconditions	A plot is stored locally	
Primary Actor	User	
Secondary Actors	Front-End	
Trigger	The user clicks the “Import Plot” button.	
Main Scenario	**Step**	**Action**
	1	Front-End prompts the user to send a plot.
	2	Front-End stores the plot inputted.
Open Issues	N/A	

Number	2	
Name	Input Parameters	
Summary	The User inputs into the Front-End the parameters of the survey being requested.	
Priority	5	
Preconditions	N/A	
Postconditions	All parameters are stored by the Front-End, ready to be sent to the Back-End	
Primary Actor	User	
Secondary Actors	Front-End	
Trigger	A new plot is requested.	
Main Scenario	**Step**	**Action**
	1	The Front-End prompts the user to send a PNG file.
	2	A PNG file is received.
	3	The Front-End changes to prompt the User for parameters.

	4	Parameters are inputted by the User.
	5	Parameters are saved by the Front-End.
Open Issues	N/A	

Number	3	
Name	Request Plot	
Summary	A plot is generated at the request to the User.	
Priority	5	
Preconditions	Parameters have been inputted by the User.	
Postconditions	A new plot has been generated by the Back-End, to be displayed.	
Primary Actor	Front-end	
Secondary Actors	Back-end	
Trigger	The User clicks the "New Plot" button.	
Main Scenario	**Step**	**Action**
	1	Inputted parameters are interpreted by the Back-End.
	2	The desired plot is generated by the backend.
Extensions	**Step**	**Branching Action**
	1a	The information given to generate the plot is invalid**:** An error message is displayed to the User describing the error.
Open Issues	N/A	

Number	4	
Name	Display Plot	
Summary	A plot has been generated and it will be displayed for the user	
Priority	4	
Preconditions	The backend has generated a plot to be displayed.	
Postconditions	The plot will be displayed to the user.	
Primary Actor	Front-End	
Secondary Actors	N/A	
Trigger	The user requests a plot	
Main Scenario	**Step**	**Action**
	1	The user requests a plot from local storage.
	2	The plot is then placed in the plot region of the GUI.
Extensions	**Step**	**Branching Action**
	1a	The desired plot does not exist in local storage**:** An error message is displayed to the User describing the error.
Open Issues	N/A	

Number	5	
Name	Compare Plots	
Summary	The user takes two plots that have been generated and views them side by side for comparison	
Priority	3	
Preconditions	The user has generated at least two plots.	
Postconditions	The GUI will show two plots side by side.	
Primary Actor	User	
Secondary Actors	Front-end	
Trigger	The user clicks the “Compare plots” button	
Main Scenario	**Step**	**Action**
	1	The application exports a .png photo of the plot to the directory which contains the application.
Open Issues	N/A	

Number	6	
Name	Save Plot	
Summary	A plot is saved locally for later retrieval	
Priority	4	
Preconditions	The backend has generated a plot the User wishes to save.	
Postconditions	The plot will be saved and stored locally to a location chosen by the User.	
Primary Actor	User	
Secondary Actors	Front-End	
Trigger	The user clicks a “Save Plot” button.	
Main Scenario	**Step**	
	1	A prompt appears prompting the User to where the plot should be saved.
	2	The Plot is saved in the location specified.
Open Issues	N/A	

2.2 Test Cases

Test Cases outline our expected measure of success for the use cases above.

- Test Case 1.1
  - A correct plot is imported into the GUI.
  - The plot is saved locally through the GUI.
- Test Case 2.1
  - Correct input is inputted into the GUI.
  - The input is saved in a way to be read by the Back-End.
- Test Case 3.1
  - Correct input is supplied to the Back-End.
  - The input is used to generate a plot.
- Test Case 3.2
  - Incorrect input is supplied to the Back-End
  - An error message is generated to inform the user of an error.
- Test Case 4.1
  - An existing plot is requested to be displayed.
  - The plot is displayed through the GUI.
- Test Case 4.2
  - A plot that does not exist is requested to be displayed.
  - An error message is generated to inform the user of an error.
- Test Case 5.1
  - Two existing plots are requested to be displayed.
  - A comparison of the two plots is generated and displayed to the GUI.
- Test Case 6.1
  - A plot is requested to be saved to a local file location.
  - The plot is saved correctly, able to be loaded by the GUI.

3. Non-Functional Requirements

Non-functional requirements were derived from the user stories agreed upon between the team and customer. They will ensure the reliability, performance, and availability of the software system.

- NR0
    - Priority: 5
    - The program shall work without an internet connection.
    - Tests for system and acceptance testing
        - SD0.1
            - Produce a graph while disconnected from all networks.
- NR1
    - Priority: 5
    - Data entered by the user shall be displayed accurately within the GUI.
    - Tests for system and acceptance testing
        - SD1.1
            - Upon importing data from a file, the data fields should accurately represent that data.
- NR2
    - Priority: 5
    - The source code for the project must be open source under an appropriate license.
    - Tests for system and acceptance testing
        - SD2.1
            - The repository containing all requisite source code will be accessible while not logged into GitHub.
- NR3
    - Priority: 4
    - 90% of users shall be able to generate a graph with less than an hour of training.
    - Tests for system and acceptance testing
        - SD3.1
            - Testing can be done on random participants who have never used the software before. They should be able to generate a graph in less than 10 minutes after no more than 50 minutes of instruction.
- NR4
    - Priority: 5
    - The software should be platform independent.
    - Tests for system and acceptance testing
        - SD4.1
            - Testing can be done by running the program on different operating systems (Windows 10, Mac OS, Linux).
- NR5
    - Priority: 5

    - The software should be package independant.
    - Tests for system and acceptance testing
        - SD4.1
            - Testing can be done by downloading the software onto a computer and have no required installations for the program to execute.
- NR6
    - Priority: 4
    - Performance should maintain visual fidelity for at minimum 95% of all user interactions.
    - Tests for system and acceptance testing
        - SD5.1
            - During regular use, the GUI should respond to all user input (clicks, button presses, and keyboard input) given a reasonably average use length.
- NR7
    - Priority: 4
    - The software should be able to process requests for up to ten plots.
    - Tests for system and acceptance testing
        - SD6.1
            - Test by inputting information for 10 distinct plots and making verifying that the resultant plots are produced.
- NR8
    - Priority: 5
    - The users should never be allowed to update the default material tensors.
    - Tests for system and acceptance testing
        - SD8.1
            - Testing can be done by running the program and manually entering the tensor data then restart the program and see if the default value has changed.
- NR9
    - Priority: 5
    - The program will reliably export data 99% of the time for all exports.
    - Tests for system and acceptance testing
        - SD9.1
            - This can be tested by running a minimum of 20 exports to ensure more than 99% of exports succeed.
- NR10
    - Priority: 3
    - The programs' source code shall be organized in compliance with Git flow to ensure outside collaborators can easily access the code.
    - Tests for system and acceptance testing
        - SD10.1
            - Git repository must be validated with proper git flow branching upon each minor release.

4. User Interface

See “User Interface Design Document for Seismic-radar toolbox GUI.”

5. Deliverables

All deliverables will be posted on Team Penobscot's Github Repository by the stated delivery date. The stated dates are set deadlines by Professor Yoo. The dates for the administrator and user manuals have not yet been announced as they are artifacts of the spring component of this course. The final project report, source code, and executable have been set to the last day of classes of the spring 2020 semester. Any deadlines listed for May of 2020 are subject to change at the discretion of Dr. Yoo. Any changes will be announced sometime in January of 2020 and will not change the dates by more than two weeks.

Deliverable	**Format**	**Delivery Date**
System Requirements Specification	PDF	October 24, 2019
System Design Documents	PDF	November 12, 2019
User Interface Design Document	PDF	November 26, 2019
Critical Design Review	Presentation	December 12, 2019
Critical Design Review Document	PDF	December 19, 2019
Administrator Manual	PDF	TBD
User Manual	PDF	TBD
Final Project Report	PDF	May 1, 2020
Source Code	Python Files (.py)	May 1, 2020
Finished Executable	Standalone Executable (.exe, .app, and linux binary)	May 1, 2020

6. Open Issues

Issues that have been raised and do not yet have a conclusion. These issues will be addressed later in the development process.

GitHub Issue Number	Short Description	Expected Resolution Date
#1	Software Requirement Specification	October 24, 2019

Appendix A – Agreement Between Customer and Contractor

By signing below, the team members understand and agree to complete the functional and non-functional requirements listed and the delivery of the deliverables listed above. By signing below, the customer understands and agrees that the completion of the functional and non-functional requirements listed and the delivering of the deliverables listed above is equivalent to completion of the software project as a whole.

Should one or both parties desire a change to this document, a new draft of this document will be formed. This new draft will be written through collaboration between both parties, to elicit shortcomings of previous versions of this document and new functional and/or non-functional requirements. This new version will be signed and dated, replacing the previous version of this document only when both parties have signed and dated the document. Should signature dates occur on different days, the new version of the document will replace the previous version on the latest date specified below.

Team Member #1:

Name (Printed): _____ Date: _____
Signature: _____

Team Member #2:

Name (Printed): _____ Date: _____
Signature: _____

Team Member #3:

Name (Printed): _____ Date: _____
Signature: _____

Team Member #4:

Name (Printed): _____ Date: _____
Signature: _____

The Customer:

Name (Printed): _____ Date: _____
Comments:_____

____

Signature: ____

Appendix B – Team Review Sign-off

By signing below all team members agree they have reviewed this document. Signing below, team members agree to all content in this document aside from any comments in the space provided below. Signing below, team members agree that the format used in this document is agreeable aside from any comments in the space provided below. Singers acknowledge that the comment area below is not a place to voice major points of contention, only minor points of disagreement in this document.

Team Member #1:

Name (Printed): _____ Date: _____
Comments:_____
_____

Signature: _____

Team Member #2:

Name (Printed): _____ Date: _____
Comments:_____
_____

Signature: _____

Team Member #3:

Name (Printed): _____ Date: _____
Comments:_____
_____

Signature: _____

Team Member #4:

Name (Printed): _____ Date: _____
Comments:_____
_____

Signature: _____

Appendix C – Document Contributions

- Adam - 30.0%
    - Generation of user stories used to generate requirements
    - Refined user stories used to generate requirements with client
    - Functional Requirements
    - Appendix A
    - Appendix B
    - Review and approval of entire document
- Alex - 25.0%
    - Generation of user stories used to create requirements
    - Refined user stories used to generate requirements with client
    - Introduction (1.1 and 1.3)
    - Nonfunctional Requirements
    - Deliverables
    - Review and approval of entire document
- Nathan - 25.0%
    - Generation of user stories used to create requirements
    - Refined user stories used to generate requirements with client
    - Nonfunctional Requirements
    - Team management
    - GitHub issue management
    - Review and approval of entire document
- Jens - 20.0%
    - Generation of user stories used to create requirements
    - Introduction (1.2 and 1.4)
    - Functional Requirements
    - Review and approval of entire document
    - Delivery of this document to customer