Project Title: System Verification and Validation Plan for Flick Picker

Author Name

November 2, 2022

Revision History

Table 1: Revision History

Date	**Developer(s)**	**Change**
October 31	Jarrod Colwell	Summary & Objectives content added

Contents

List of Tables

[Remove this section if it isn't needed —SS]

List of Figures

[Remove this section if it isn't needed —SS]

1 Symbols, Abbreviations and Acronyms

Symbol	Description
T	Test
V&V	System Verification and Validation
SRS	Software Requirements Specification
UI	User Interface

[symbols, abbreviations or acronyms – you can simply reference the SRS (Author, 2019) tables, if appropriate —SS]

This document ... [provide an introductory blurb and roadmap of the Verification and Validation plan —SS]

2 General Information

2.1 Summary

This document describes the plan to verify and validate that Flick Picker meets the defined requirements and specifications. Additionally, this document will validate that Flick Picker fulfills its intended purpose of recommending compatible Movies, TV Shows, or Anime to an individual or a group.

2.1.1 Front End Testing

- Account Creation - Web page that facilitates user account creation
- User Preferences - Web page that facilitates user preference settings
- Group Creation - Web page that facilitates the creation of groups
- Recommendation - Web page that displays recommendations for an individual or group

2.1.2 Back End Testing

- Database Access - Accessing the database to find user preferences or information pertaining to Movies, TV Shows, or Anime
- Recommendation Algorithm - The algorithm responsible for finding the best Movies, TV Shows, or Anime for an individual or group
- API Data - The accessing, storage, and usage of external data from various APIs

2.2 Objectives

2.2.1 Requirements

The first objective involves verifying that Flick Picker meets requirements outlined in our SRS document and validating that the behaviour present is

desirable. This includes the functional requirements (e.g. Authentication Requirements) and the non-functional requirements (e.g. Appearance Requirements). This objective will build confidence in the correctness of Flick Picker along with validating that security and usability standards are met.

2.2.2 UI Elements

The second objective of the V&V document involves the validation of navigability and functionality of UI elements. Each menu described in the 'Front End Testing' section above must be functional for users. Additionally, users must be able to navigate to each page individually. This objective ensures that usability and response time standards are met.

2.2.3 External Connections

The final objective of the V&V document is to ensure that all external connections of Flick Picker (e.g. APIs, Database) work as intended. This objective ensures that the interoperability of Flick Picker is adequate.

2.3 Relevant Documentation

[Reference relevant documentation. This will definitely include your SRS and your other project documents (MG, MIS, etc). You can include these even before they are written, since by the time the project is done, they will be written. —SS]

Author (2019)

3 Plan

[Introduce this section. You can provide a roadmap of the sections to come. —SS]

3.1 Verification and Validation Team

[You, your classmates and the course instructor. Maybe your supervisor. You shoud do more than list names. You should say what each person's role is for the project. A table is a good way to summarize this information. —SS]

3.2 SRS Verification Plan

[List any approaches you intend to use for SRS verification. This may just be ad hoc feedback from reviewers, like your classmates, or you may have something more rigorous/systematic in mind.. —SS]

[Remember you have an SRS checklist —SS]

3.3 Design Verification Plan

[Plans for design verification —SS]

[The review will include reviews by your classmates —SS]

[Remember you have MG and MIS checklists —SS]

3.4 Implementation Verification Plan

[You should at least point to the tests listed in this document and the unit testing plan. —SS]

[In this section you would also give any details of any plans for static verification of the implementation. Potential techniques include code walkthroughs, code inspection, static analyzers, etc. —SS]

3.5 Automated Testing and Verification Tools

[What tools are you using for automated testing. Likely a unit testing framework and maybe a profiling tool, like ValGrind. Other possible tools include a static analyzer, make, continuous integration tools, test coverage tools, etc. Explain your plans for summarizing code coverage metrics. Linters are another important class of tools. For the programming language you select, you should look at the available linters. There may also be tools that verify that coding standards have been respected, like flake9 for Python. —SS]

[The details of this section will likely evolve as you get closer to the implementation. —SS]

3.6 Software Validation Plan

[If there is any external data that can be used for validation, you should point to it here. If there are no plans for validation, you should state that here. —SS]

4 System Test Description

4.1 Tests for Functional Requirements

[Subsets of the tests may be in related, so this section is divided into different areas. If there are no identifiable subsets for the tests, this level of document structure can be removed. —SS]

[Include a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good. —SS]

4.1.1 Area of Testing1

[It would be nice to have a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good. If a section covers tests for input constraints, you should reference the data constraints table in the SRS. —SS]

Title for Test

1. test-id1

   Control: Manual versus Automatic

   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

   Control: Manual versus Automatic

   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:

4.1.2 Area of Testing2

...

4.2 Tests for Nonfunctional Requirements

[The nonfunctional requirements for accuracy will likely just reference the appropriate functional tests from above. The test cases should mention reporting the relative error for these tests. —SS]

[Tests related to usability could include conducting a usability test and survey. —SS]

4.2.1 Area of Testing1

Title for Test

1. test-id1

   Type:

   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:

4.2.2 Area of Testing2

...

4.3 Traceability Between Test Cases and Requirements

[Provide a table that shows which test cases are supporting which requirements. —SS]

5 Unit Test Description

[Reference your MIS and explain your overall philosophy for test case selection. —SS] [This section should not be filled in until after the MIS has been completed. —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 Look and Feel Tests

Test #2: Ease of Use Test

Control: Non-Functional

Initial State: System is in the login screen state

Input: User logs in and explores the different screens and services offered

Output: User goes through the system and provide feedback on the style choices (such as size of buttons or overall colour palette)

Derivation: The test is meant to assess how different users may feel about style decisions and offer feedback on what looks and feels good to them and what needs improvement

Execution: Survey

5.3.2 Usability and Humanity Tests

Test #3: Ease of Use Test

Control: Non-Functional

Initial State: System is in the login screen state

Input: User logs in and explores the different screens and services offered

Output: User goes through the system without needing to ask for assistence or misunderstanding the affordances of the different screens

Derivation: The test case is meant to assess whether the presentation of the system informs users of different technological skill levels of what functionality is provided by whatever state/screen they may be on

Execution: User Test

Test #4: Learning Test

Control: Non-Functional

Initial State: Users have just finished Test #3

Input: Users are given a survey of how easy/complicated their experience using the system was

Output: The surveys are filled out and collected

Derivation: This survey is a continuation of Test #3, with its results providing feedback as to whether the GUI of the system is understandable to users of different skill levels

Execution: Survey

5.3.3 Operational and Environmental Test

Test #12: Test for Interfacing with Adjacent Systems

Control: Manual

Initial State: System is in the login menu state

Input: Developer will use the system as a regular user would on different web browsers

Output: The system will have the same fundamental functionalities on all browsers, with perhaps minor differences in visual presentation

Derivation: Developer will go through the various functionalities of the system on different web browsers, such as Chrome and Firefox, to assess whether or not it works and is usable on each

Execution: Developer Exploration Test

5.3.4 Maintainability and Support Tests

Test #13: Test for Adaptability

Control: Manual

Initial State: System is in the login menu state

Input: Developer will use the system as a regular user would on different mobile devices

Output: The system will have the same fundamental functionalities on different mobile devices, with perhaps minor differences in visual presentation

Derivation: Developer will go through the various functionalities of the system to assess whether or not it works and is usable on each

Execution: Developer Exploration Test

5.3.5 Security Tests

Test #14: Test for Access

Control: Manual

Initial State: System is in the main menu state

Input: Developer will click on their user profile and test the ability to change settings, as well as view the profile information of other users and ensure they cannot change their settings

Output: The Developer's user profile will be changed, with such changes being visible to other users (if the setting was one which can be viewed by others) with the content of other profiles being unchangeable

Derivation: Developer will ensure that the user's data is changable, that changes to public information can be viewed by others, that others' settings cannot be changed, and private information cannot be viewed by other users

Execution: Developer Exploration Test

Test #15: Test for Integrity

Control: Manual

Initial State: System is in the main menu state

Input: Developer will change a profile or preference setting

Output: After different increments of time after the changes are made (such as immediately, 1 hour, and 24 hours after) the Developer will verify that all the changes made persist and do not revert to a prior state

Derivation: This is the best way to verify that information persists and retains its integrity over time, with it being possible to infer future persistence of information after the testing period

Execution: Developer Exploration Test

Test #16: Test for Privacy There is no possible test for this requirement, as it pertains to a regulation on how users' information may be used by those with access to it.

5.4 Traceability Between Test Cases and Modules

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

References

Author Author. System requirements specification. https://github.com/..., 2019.

6 Appendix

This is where you can place additional information.

6.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

6.2 Usability Survey Questions?

[This is a section that would be appropriate for some projects. —SS]