|  |
| --- |
|  |
|  |  |

**Professor Panic: The Academic Exodus**

**Verification and Validation Plan**

Version 2.0

12/01/2023

|  |
| --- |
|  |
|  |  |

**Professor Panic: The Academic Exodus**

**Risk Management Plan**

**12/01/2023**

**WESTERN NEW ENGLAND UNIVERSITY**

**CPE -425, 525 SOFTWARE ENGINEERING**

**Team#1 Project**

DOCUMENT APPROVALS

|  |  |  |
| --- | --- | --- |
| **NAME** | **SIGNATURE** | **DATE** |
| David Melanson (Project Lead) |  |  |
| Kalyan Bhagavan |  |  |
| Matthew Dingman |  |  |
| Philip Brehart |  |  |
| Logan Paranto |  |  |
| Sydney Adams-Jones |  |  |

DOCUMENT CONTROL / REVISION HISTORY

|  |  |
| --- | --- |
| **TITLE** | Verification and Validation Plan |
| **REPOSITORY LOCATION** | GitHub |
| **URL** | https://github.com/PJB01/Academic-Exodus |

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 10/02/2023 | 1.0 | Initial Draft | KB |
| 11/28/2023 | 1.1 | Minor revisions, some descoping and tailoring | DM |
| 12/01/2023 | 2.0 | More descoping and tailoring | ALL |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

Contents

[SECTION 1. OVERVIEW](#page4) [2](#page4)

[1.1](#page4) [Scope](#page4) [2](#page4)

[1.2](#page4) [V&V Objectives](#page4) [2](#page4)

[SECTION 2. MANAGEMENT APPROACH](#page6) [4](#page6)

[2.1](#page6) [Responsible teams](#page6) [4](#page6)

[2.2](#page7) [Modification of activities](#page5) [5](#page7)

[2.3](#page7) [Training](#page5) [5](#page7)

[2.4](#page7) [Development/testing framework](#page5) [5](#page7)

2.5 [Verification](#page5) Requirements [5](#page7)

[SECTION 3. TESTING SCHEDULE](#page8) [6](#page8)

[3.1](#page8) [Testing Schedule](#page8) [6](#page8)

[3.2](#page10) [Equipment required](#page10) [8](#page10)

[3.3](#page10) [Software required](#page10) [8](#page10)

[3.4](#page10) [Personnel required](#page10) [8](#page10)

[3.5](#page10) [Deliverables](#page10) [8](#page10)

[3.6](#page10) [Testing tools](#page10) [8](#page10)

[3.6](#page10) [Site supplemental materials](#page10) [8](#page10)

[SECTION 4. TEST REPORTING](#page11) [9](#page11)

[4.1](#page11) [Template and policy for recording tests](#page11) [9](#page11)

[4.2](#page11) [Policy for failed tests and anomalies](#page11) [9](#page11)

**SECTION 1. OVERVIEW**

**PURPOSE**

The purpose of this Software Verification and Validation (V&V) Plan is to provide a comprehensive framework for ensuring the successful development, testing, and release of **The Professor Panic: The Academic Exodus**. This plan outlines the methodologies, procedures, and criteria for verification and validation activities, which are critical to achieving a high-quality, functional, and reliable game.

The Academic Exodus is an interactive software application designed to provide players with an engaging gaming experience from a top-down perspective. It encompasses various game elements, including player controls, non-player character behavior, game mechanics, graphics, sound, and overall gameplay. The purpose of this V&V plan is to define the structured processes required to ensure that these elements align with the defined requirements and meet the expectations of both the development team and end-users.

**1.1** **SCOPE**

The scope of this V&V plan encompasses the entire development and testing lifecycle of the Academic Exodus. It begins from the initial design and development stages, continues through extensive verification and validation activities, and concludes with the final release of the game to end-users.

**1.2** **V&V OBJECTIVES**

The objectives of Verification and Validation (V&V) in software development are critical for ensuring that the software meets its requirements, functions correctly, and provides a high level of quality. These objectives help in delivering a reliable and effective software product. The main V&V objectives:

* Requirements Confirmation: Verify that the software meets specified requirements.
* Functionality Validation: Ensure the software functions correctly and aligns with user needs.
* Defect Identification: Detect and address software issues or bugs.
* Quality Assurance: Build high-quality software with adherence to standards.
* Reliability Enhancement: Improve software reliability and stability.
* User Acceptance: Confirm the software meets user expectations.
* Performance Evaluation: Assess software performance under various conditions.
* Documentation and Reporting: Maintain clear records of V&V activities.
* Risk Mitigation: Mitigate risks associated with software development.
* Release Decision: Provide information for informed release decisions.
* Continuous Improvement: Use insights for process improvement

**SECTION 2. MANAGEMENT APPROACH**

The scheduling of V&V tasks is driven by the software development schedule. Therefore, a V&V task is performed in relationship to what software development activities are taking place. One or more V&V tasks can be performed concurrently until a task is completed. A task is considered completed when the required report e.g., V&V sign-off, reports, etc. are satisfactorily completed or action items have been closed. The following tasks, requiring coordination and cooperation with the project team, shall be performed by V&V.

**2.1** **RESPONSIBLE TEAMS**

Stakeholders:

- Sponsor

- Customers

- Users

Development and Planning Team (Plan):

- Sponsor

- Customer

- User

- Requirements engineer/Project Manager

- Developer

- QA/Tester

Development Team (Dev):

- User

- Requirements engineer/Project Manager

- Developer

- QA/Tester

V&V Team (V&V):

- Requirements engineer

- Developer

- Quality assurance /Tester

**2.2** **MODIFICATION OF ACTIVITES**

This document details activities generally helpful for projects. V&V teams can assess what works best in their environment.

**2.3 TRAINING**

For initial training, it is recommended to consult online repositories and courses pertaining to the module ‘Pygame’. As the project moves onward, cooperation between developers will be the best source of training and knowledge. Training sessions will be held on an as-needed basis to educate the group on topics that one engineer is more experienced in when it is apparent that it will be beneficial to the success of the project. Members may be required to accomplish ‘spikes’ when the expertise necessary is not found within the group already.

**2.4 DEVELOPMENT/TESTING FRAMEWORK**

The Pygame development/testing framework comprises the Pygame library, version control (e.g., Git), an integrated development environment (IDE), continuous integration tools, defect tracking systems, and peer reviews.

**2.5 VERIFICATION REQUIREMENTS**

The plan will address the following key areas:

* **Requirements Verification**

The Software Development Plan will ensure that the game's requirements are well-defined, clear, complete, and consistent. It will encompass processes for stakeholder engagement to validate that the game's requirements align with their expectations and needs.

* **Code Review**

Code reviews will be conducted to identify and rectify coding errors, ensuring that the code adheres to industry-standard best practices. This process aims to improve code quality and maintainability.

* **Testing**

The V&V plan covers a variety of testing activities, including unit testing, integration testing, and functional testing. These tests are essential to verify that the game functions as expected and operates efficiently.

* **Documentation and Reporting**

Comprehensive documentation of test plans, test cases, and test results is an integral part of the plan. It ensures that the development team has a record of the testing process and outcomes, enabling them to make informed decisions and address issues effectively.

* **User Acceptance Testing**

To ensure that the game aligns with user expectations and preferences, user acceptance testing will be conducted. Their feedback will guide refinements and adjustments to improve the overall gaming experience.

Surveys will be conducted to document opinions and suggestions for utilization in improvements to software.

**SECTION 3. TESTING SCHEDULE**

**3.1** **TESTING SCHEDULE**

The master schedule summarizes the various verification and validation tasks and their relationships to the overall project. Describe the project life cycle and project milestones including completion dates. Summarize the V&V tasks and how results provide feedback to the development process to support overall project management functions.

Sprint tests occur after the sprint is completed by the V&V team (including the tester on the development teams) - Additional business, user interface, external interface, component tests, quality attribute tests, and exploratory testing.

For the above activities prepare a schedule for this project. Identify who will perform the activities, date when the activity will be completed, and contingencies should responsible individuals need to be replaced.

**Deliverable**

**Responsibility**

**Due date**

**3.2 EQUIPMENT REQUIRED**

For project development, we need a computer with sufficient hardware, a development environment (IDE or text editor), and input devices (mouse, keyboard, or game controller). Optional equipment may include graphics tablets and platform-specific testing hardware.

**3.3** **SOFTWARE REQUIRED**

We need Python and the Pygame library, along with an IDE for coding. Optional software includes image and sound editing tools, as well as version control software for collaboration.

**3.4** **PERSONNEL REQUIRED**

We required developers, game designers, testers, a project manager, Roles may overlap in smaller teams.

**3.5** **DELIVERABLES**

The key deliverables include the game code, assets, testing reports, game documentation, the playable game, marketing materials, defect reports, and source code repository.

**3.6** **TESTING TOOLS**

Most testing will be done internally during code reviews and before software is submitted. It is expected that, if software is submitted to the GitHub repository that it aligns with the current accepted version and runs defect free, to this effect, testing documentation must be completed before submission. Before integration, configuration management will be completed to ensure that this does not affect the standing version of the software. The onus of testing falls first to the developer submitting software to be integrated, secondly on the configuration manager approving the revision, and barring those, on the project lead.

**3.7 SITE SUPPLEMENTAL MATERIALS**

Essential supplemental materials include the Pygame website, community forums, official documentation, GitHub repositories, YouTube tutorials, online courses, game assets websites, the Python community, and relevant books. These resources offer valuable support and knowledge.

**SECTION 4. TEST REPORTING**

**4.1** **TEMPLATE AND POLICY FOR RECORDING TESTS**

The software development process follows Test-Driven Development (TDD) principles, where tests are created before the code and are validated by ensuring they fail first and pass after implementation.

Continuous integration is a fundamental practice, which means that with each code addition to the repository, a new build and test are automatically conducted.

Throughout the sprint cycles, the development and testing framework operates according to the above processes. Any major issues identified within this framework are logged in the test and anomaly tracker.

All user and manual tests performed subsequent to sprint development are to be recorded in the following:

**Id Date**

|  |
| --- |
| **to** |

|  |
| --- |
| **document Reference** |

|  |
| --- |
| **document within Reference** |

**Build**

**Commit**

**ID**

**Result**

**Observer**

**Comment**

**4.2** **POLICY FOR FAILED TESTS AND ANOMALIES**

Upon encountering a failed test, it is essential to promptly log an issue or defect in the defect tracker with the following details:

- Test identifier (obtained from the test reporting)

- Reference to the associated document and location within that document

- Build version number

- A clear explanation of the observed failure

- Detailed steps to reproduce the issue

- Severity level (high, medium, low)

- Priority level (high, medium, low)

- Name of the person reporting the bug

- Assigned individual/team responsible for resolution

- Date of the bug report

- The reason behind the issue

- Current status (new, open, active)

- Environment details (e.g., Windows 11, SQL Server 2007)

- Any additional notes or relevant information

In the event of a test failure, a defect review is initiated. During this review, the planning team assesses the necessity of making updates to the project's charter, features, stories, and scenarios. If such updates are deemed necessary, the team may reschedule related work accordingly.