Dual-Pi Visualizer Requirements Document

Table of Contents

1 Introduction

This document details the requirements for the Dual-Pi Visualizer project, including its purpose and scope, overview, use cases, functional and nonfunctional requirements and its projected milestones and deliverables. The introductory section details the scope of this project, as well as its purpose and target audience. A list of terms and definitions that will be used is also provided.

1.1 Purpose and Scope

The purpose of this document is to give a clear description of the requirements for the Dual-Pi Visualizer project. It explains how this application will be developed, who will be using it, and what functional and nonfunctional requirements exist. This document will be used as a reference during the future phases of development.

1.2 Target Audience

The target audience for this document are stakeholders and developers. This document is intended for all who are involved in the completion of this project, and therefore, its scope is broad.

1.3 Terms and Definitions

The following terms will be utilized throughout this document.

Term	**Definition**
User	A client that interacts with the Pi
Use Cases	The interaction between the product and the users of the product.
Stakeholder	A person that is involved in the system, that is not part of development.
Milestone	The date on which a work product will be completed.
Deliverable	A measurable outcome of the project.
GUI	Graphical User Interface
Functional Requirement	An action that the product must be able to perform.
Nonfunctional Requirement	Specifications of the properties of the product.

2 Product Overview

This section will explain the functionality of the entire project. The purpose of this section is to provide an overview of what this application must achieve, as well as who is involved as stakeholders. It will describe the different users of the application, stakeholders, and use cases.

2.1 Users and Stakeholders

This section introduces the stakeholders and users that will be utilizing and interacting with this application. It also describes the differences in functionality between stakeholder types.

2.1.1 Project Leader

The first stakeholder is the project leader. They will develop the application with the development team from the requirements phase through completion. The development team will deploy the project upon completion and test its use.

2.1.2 Project Customer

The project customer is a project user. They will utilize the completed project to display visual art at Burning Man.

2.2 Use cases

The following use cases describe the possible interactions between the external users and the project. Sequence diagrams are included.

2.2.1 Turn on a Pi

1. Starts the program
2. Determines if another Pi is the leader and if so, syncs up to it.
    a. Otherwise this Pi becomes the leader.
3. Finds the connected Projector and begins visualization.

2.2.2 The Pi is on past 7 AM

1. If it is the leader, another Pi becomes the leader.
2. Stop producing an image for this Pi
3. Turn off.

2.2.3 A projector is attached to a Pi (or will they automatically be attached?)

1. Produce current selected image.

2.2.4 User presses an input on the device (mouse?)

1. Produce desired output for input received.

2.2.5 A Pi is turned off

1. If it is the leader, another Pi becomes the leader.
2. Stop produce an image for this Pi
3. Turn off.

2.2.6 Any error has happened

1. Flash a current color depending on the error
2. Turn off if fatal

2.2.7 A pi is unplugged from the network

1. If follower, halt projection, wait for reconnection, then search for leader
2. If leader, halt projection, wait for reconnection, then search for leader

3 Functional Requirements

A functional requirement specifies an action that the product must perform. These requirements are expressed in terms of input and output. This section covers all functional user requirements.

3.1 Program starts on boot

3.1.1 Program begins automatically without user input when the RaspberryPi is powered up.

3.1.2 Program begins in full screen mode automatically when it starts.

3.1.3 If this Pi is designated the “leader” unit *(see 3.2), the program randomly populates a video playlist at boot, with both 30-60 second videos and longer movies. Otherwise, this unit is designated “follower” and copies the “leader”’s playlists.

3.1.4 The playlist of videos will begin playback automatically once the playlist has been populated, and the synchronization steps have been completed. *(see more in “3.2 Synchronization” and “3.3 Video playback”)

3.1.5 All wifi, bluetooth, and internet signals (aside from our ethernet and USB bluetooth connections) will be disabled on boot.

3.2 Synchronization

3.2.1 At startup, the RaspberryPi will check for an ethernet connection, and a designated “leader” Pi unit. If either of these conditions are not met, this unit will be designated the “leader” unit, and will signal other units of its status upon request.

3.2.2 The “leader” unit will broadcast startup timings for all modes and visualizations.

3.2.3 The “follower” unit(s) will wait for startup timings for all modes and visualizations, only initiating visual projections upon receiving that data.

3.3 Video Playback

3.3.1 Program randomly populates a video playlist at boot, with 30-60 second videos and movies.

3.3.2 The playlist of videos will begin playback automatically once the playlist has been populated.

3.3.3 The movie files will skip the credits in the beginning and end. Videos of an hour or more the first 15min and last 18min will be cut out.

3.3.4 Between the playback of each longer video, ProjectM for VLC will display visuals, using locally stored music files for tempo and intensity.

3.3.5 At the end of video playlist, the Pi unit will shuffle the videos and restart the playback unless user switches the mode.

3.3.6 All videos are stored in the sd card in a single video folder.

3.4 Control Schematics, User Access

3.4.1 The “leader” Pi unit can be accessed by a user from controls on a wireless mouse, with its usb transmitter connected to that microcomputer.

3.4.2 The controls are as follows:

Left Click – Next Visual/video, Exit Right click

Double Left Click 1 – Minimize Program

Double Left Click 2 – Maximize Program

Right Click – Keep Current Visual for 60 Minutes (If video playing is less than 60 minutes long, loop)

Wheel Up – Speed up visualization playback

Wheel Down – Slow Down visualization playback

Wheel Click – Return to normal playback speed

3.5 Power Down

3.5.1 Every unit (“leader” and “follower”) will automatically check if the time is between 7am and 7pm. If it is, the unit will automatically power itself down.

3.6 Additional Enhancements Possible

3.6.1 Audio input affected ProjectM timing/intensity of visualizations
3.6.2 Audio input controls timing of next visualizations start
3.6.3 Control of LED lights in synch with the visuals
3.6.4 Audio input affects LED lights
3.6.5 Other open-source visualizers
3.6.6 Flamethrowers!!
3.6.7 Wireless keypad inputs
3.6.8 More than a single “follower” unit hooked into the network
3.6.8 Keyboard input that affects the visuals - Hue change and intensity

4 Nonfunctional Requirements

A nonfunctional requirement specifies the properties of the target product, rather than an action to be performed. Nonfunctional requirements deal with the operation of the system itself.

4.1 Reliability

The project will run correctly over 99.95 percent of the time.

4.2 Response Time

This project will have a fast response time.

4.3 Storage

Video and visualizer storage must be easy to I/O

4.4 Durability

The computers and projectors will have effective heat dissipation, protection from weather hazards.

4.5 Open source

This project will be freely available to use by anyone. We will need to choose which license to attach to our project. MIT license.

4.6 Deployment

Deployment of this system will be simple enough for use by average users.

4.7 Extensibility

Features may be added to the project platform with minimal complication through version updates and/or users.

4.8 Removes Bloatware

5 Milestones and Deliverables

The purpose of this section is to detail the timeline and milestones of this project. The milestones begin with the design phase, including system architecture and providing a detailed system design, followed by creating a test plan, and then programming the application. These milestones will follow each other in succession, although there will be some crossover between them.

5.1 Design

This milestone consists of developing the design for this application. The deliverable that is produced at this milestone is the Design Document. The work involved to reach this milestone includes determining the constraints and dependendencies of this project, as well as determining the methodology of the design. It also includes high-level designing the entire project in order to be fully prepared to begin programming. The system architecture must be determined, as well as all important algorithms and diagrams. Furthermore, a detailed system design must be determined with data structures included.

5.1.1 Design - System Architecture

This stage involves designing and providing a detailed description of the system architecture, including subsystems, components, and objects of the system. Important algorithms will be included as well. A dataflow diagram will also be produced as a deliverable.

5.1.2 Design - Detailed System Design

This stage involves designing and explaining the data structures and functions that will be necessary. Detailed class diagrams will be produced as deliverables.

5.2 Test Plan

This milestone consists of developing a test plan for the application. The deliverable that will be produced at this milestone is the Test Plan Document. This stage involves determining unit testing, as well as integration testing for the application.

5.3 Programming

The end goal of this milestone is the completed application. This section will be split into many deliverables with projected due dates.

~~Secret text~~