Digital Signage

by Team JEDi

Jared Miller

Ariunbold Batsaikhan (Eric)

Darien Clay

SLACK CHANNEL: [csc483jedi.slack.com](https://csc483jedi.slack.com/)

Final Document

For: Dr. Allison

On: Winter 2018

**Table of Contents**

[**Abstract**](#_7484k3xk2pe4)

[1.1. Purpose of System](#_ea3c7paqdoh7)

[1.2. Scope of System](#_n3317s445qt3)

[1.3. Development Methodology](#_eoy3h5prfdsl)

[1.4. Definitions, Acronyms, and Abbreviations](#_me8lhhsm816v)

[1.5. Overview of Document](#_iogr7fhnx5ik)

[**2. Current System**](#_b50vlnp6jqux)

[**3. Project Plan**](#_1t3h5sf)

[3.1. Project Organization](#_4d34og8)

[3.2. Software and Hardware Requirements](#_2s8eyo1)

[3.3. Work Breakdown](#_nn2omljbf51t)

[**4. Requirements of System**](#_3rdcrjn)

[4.1. Functional and Nonfunctional Requirements](#_l3k46gc3qfvc)

[4.1a Functional Requirements](#_kypj0yo90433)

[4.1b Nonfunctional Requirements](#_wj83zksn2vbb)

[4.2. Identified Personas](#_7ltqssgz15hf)

[4.3. Use Case Diagram](#_sagdxdhwmwmt)

[Actor Descriptions](#_gco6cmf79rr)

[**5. Software Architecture**](#_44sinio)

[5.1. Overview](#_g0ats0gppztm)

[5.2. Subsystem Decomposition](#_z337ya)

[5.3. Persistent Data Management](#_3j2qqm3)

[Management Interface](#_2b1o3e3lbsmv)

[Player System](#_l2mb7l8tdypc)

[**6. Object Design**](#_papcedbif4t4)

[6.1. Overview](#_6jsucqwtat1s)

[6.2. Object Interaction](#_2xcytpi)

[6.3. Detailed Class Design](#_ljnfty6x28us)

[**7. Testing Process**](#_3whwml4)

[**8. Glossary**](#_xv52gwsxepe6)

[**9. Appendix**](#_qsh70q)

[9.1. Appendix A - Use Cases](#_ct5ctcgo9fo1)

[9.1. Appendix B – User Interface Designs](#_jx7pws138sdc)

# **Abstract**

This document describes the scope and development plan for a digital signage system to be created by “Team JEDi” of the Winter 2018 section of Information Systems Design Project directed by Dr. Mark Allison at UM-Flint. The system is intended for use in information distribution (e.g. upcoming events, club meetings) and advertising via television units installed at a site (e.g., in a university building). The system is an attempt to improve upon existing solutions in the areas of usability, ease of installation, hardware footprint and energy efficiency.

The management portion of the system be a web based solution, with player systems based off Raspberry Pi 3 devices.**1. Introduction**

In this chapter we will introduce the motivation for building the system

## ***1.1. Purpose of System***

The purpose of the Signage system is to allow organizations to create dynamic displays for information or marketing purposes. The system attempts to improve on the ease and cost of installation, usability of management interfaces, and energy footprint of existing solutions.

Currently at UM-Flint there exists signage at various locations providing event information, marketing for student groups, pathfinding and weather info. The current solution requires a full Windows computer to run each sign, bringing with it all the requirements of licensing and maintaining a Windows system. The signs are also relatively open to human interference in the form of unauthorized persons attaching a keyboard/mouse to the computer and performing other actions. The computers also require network connectivity, necessitating access to an ethernet interface (expensive to run new lines) or wireless access (not native to most desktop hardware).  
 To provide an alternative solution, we propose the creation of a system fulfilling a similar role, but able to run on cheaper hardware with software created solely for the purpose of signage. The main candidate platform is the Model B Raspberry Pi 3, a low-cost SoC computer featuring a native 802.11 wireless interface, HDMI output (usable on most display platforms), and a sane programming environment allowing easy duplication simply by cloning an SD card.

## 1.2. Scope of System

The system will be comprised of two primary components: a management interface and a player system. The management system will allow users to upload, create layouts for, and schedule content. The player system will receive the content and metadata from the management system and display it on a standard TV or monitor interface. There will exist capabilities for the following high level features:

* Static image content
* Data feeds (e.g. weather)
* Division of screen space, each section containing independant content
* Ability to deploy content wirelessly, over a standard 802.11/WiFi network

## ***1.3. Development Methodology***

The project will follow a roughly Scrum-ian methodology. The project will be initialized by listing features with a potential to be included in the project, ranked by importance. Work will be divided into sprints, during which the following cycle will be performed:

1. Select most important features to be implemented
2. Assign tasks based on selected features
3. Meet to discuss progress or seek assistance/guidance
4. Finish sprint with a viable implementation

## ***1.4. Definitions, Acronyms, and Abbreviations***

Actors:External entities that interact with the system.

Git: A source control solution, housing code for the product

Management Interface: The component of the product with which the user interacts. Accepts media and layouts, deploys code to players

Player: The component of the system responsible for receiving and displaying media.

DD:Design Document.

Deliverable:Work product for client.

JS: JavaScript, or ECMAScript, a programming language

Raspberry Pi: A SoC based computer featuring low cost, low energy consumption, and small footprint.

SoC: System on a chip

## ***1.5. Overview of Document***

The remainder of the document details schedule/resource planning and high level project design details.

# 

# **2. Current System**

This system is intended to fulfill many of the objectives of the existing Four Winds Interactive Content Manager solution. The current system is employed by UM-Flint to manage several signs on campus and features content scheduling, layout, interactivity and wayfinding features.

# 

# **3. Project Plan**

## ***3.1. Project Organization***

**Implementation:** Jared

**Design:** Jared, Eric

**Documentation:** Jared, Eric, Darien

**Testing:** Eric

## 

## ***3.2. Software and Hardware Requirements***

Hardware: laptop/PC (i3 or newer CPU, 8 GB of RAM, 20 GB free of hard disk space, CD drive).

Software:Chrome/Firefox/etc, StarUML, NodeJS, VSCode/Sublime/etc, Git ("Git for windows" recommended), SSH client (ssh, PuTTY).

## 

## ***3.3. Work Breakdown***

| **Task #** | **Task** | **Description** | **Duration** | **Dependencies** |
| --- | --- | --- | --- | --- |
| 1 | Project Plan | Get to know team members, brainstorm, assign roles, decide project topic | 3 days |  |
| 6 | Create use cases | Identify use cases, assign use cases to team members, each team member develops their assigned use cases | 8 days | 1 |
| 10 | Review and completion of use cases | Present use case diagrams to professor, correct use cases | 5 days | 6 |
| 13 | Development of Personas | Complete | 12 days | 10 (M1) (D1) |
| 17 |  | Present SRD to class, submit SRD to professor | 1 day | 13 |
| 18 | Software Architecture | Divide project into subsystems, identify objects, complete design document, | 2 days | 17 |
| 22 | Object Design | Transition of software models into source code. | 18 days | 17 |
| 30 | Implementation | Database design. Interface Layer, Application Layer and Storage Layer coding. | 40 days | 27 (M3) |
| 35 | Testing Process | Subsystem, System and Evaluation tests. Creation of the User’s Guide. | 16 days | 27 |
| 40 | Creation of FD | Complete FD, create Power Point presentation. | 23 days | 30, 35  (M4) (D3) |
| 45 | Presentation of FD | Present and submit the FD. | 1 day | 40 |

M – Milestone D – Deliverable

# **4. Requirements of System**

The system we are proposing

## ***4.1. Functional and Nonfunctional Requirements***

### 4.1a Functional Requirements

* Player system shall receive content and metadata over a network connection
* Player system shall be capable of rendering content over an HDMI
* Management interface shall be capable of being served over HTTP to standard web browsers
* Management interface shall be capable of receiving standard image media files
* Management interface shall be capable of deploying content to the player system over a standard TCP/IP network connection
* System shall allow for scheduling of content between particular dates
* System shall allow for password based authentication to the management interface

### 4.1b Nonfunctional Requirements

* Player system shall have a minimum time between failure of 1 day (24 hours) (reliability)
* Player shall not display any observable stutter/breakage in rendering for content transitions for at least 95% of the time (reliability)
* Other actors should be unable to add content or metadata to the players without authorization (security)
* Content being sent to players should be unreadable by any observers (security)
* A new person should be able to upload a media file no more than five minutes with documentation. (Usability)

## ***4.2. I***dentif***ied Personas***

* Individuals setting up new players
  + Mike the Installation guy sets up signs on campus. Every so often there will be a desire for a new sign in a new building or department, and it falls to Mike to obtain the hardware, configure it and install it jat the new site. Mike doesn’t want to spend a long time figuring out obscure details of the system to get it running/licensed. Mike enjoys one clear path to follow, described in documentation.
* Individuals submitting media for display
  + Janet the head of basket weaving club wants to upload a picture on the campus tv’s to get people to come to the basket weaving club. Janet does not want to spend a lot of her free time figuring out another system of how to upload advertisement of her basket weaving club. Janet doesn’t use technology a lot so she relies on the IT department to help with uploading her advertisement pictures.
* Individuals managing layouts and schedules for signs
  + Bob the IT department office worker wants to unify and set up proper timing for all signage and posters of the campus. Bob is collecting all needed information for different infrastructures of signages. Bob enjoyed simple implementation of our project.

## ***4.3. Use Case Diagram***

The next figures depict the interaction between the actors and the previously described use cases. A description for each actor follows. The human actors (left) are labelled by their “roles” or permissions in the system. Detailed text use cases can be found in Appendix A.

### Actor Descriptions

**Any User**: A human actor using the management interface, may have other permissions.

**User Manager Role:** A human actor with permissions to manage users: add/remove users and alter their permissions.

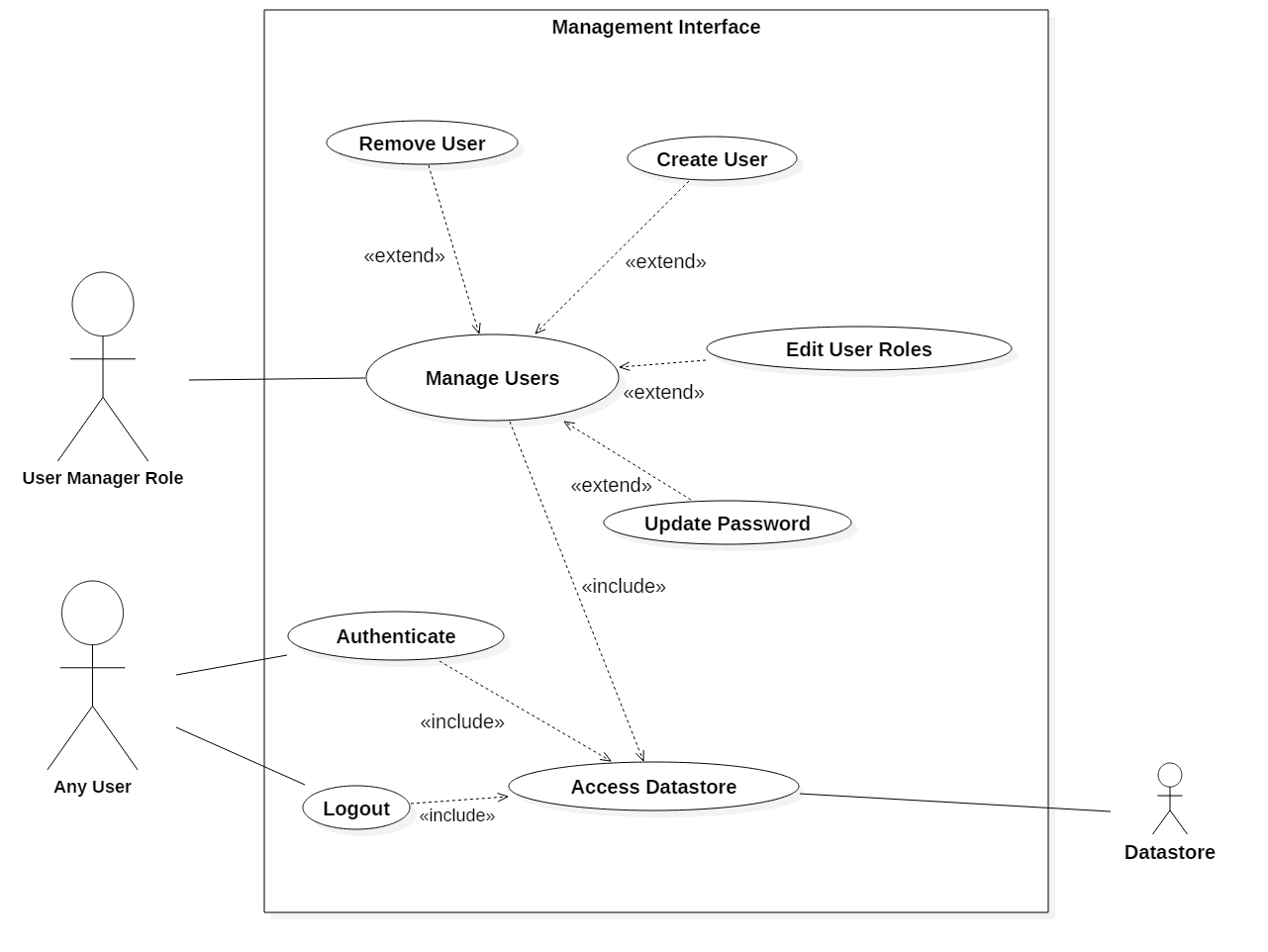
**Media Role:** A human actor with permissions to upload and edit media file information.

**Layouts Role:** A human actor with permissions to create and edit layouts.

**Player Role:** A human actor with permissions to create, edit, and deploy players.

**Datastore**: Storage on the management interface containing the database in addition to uploaded media files

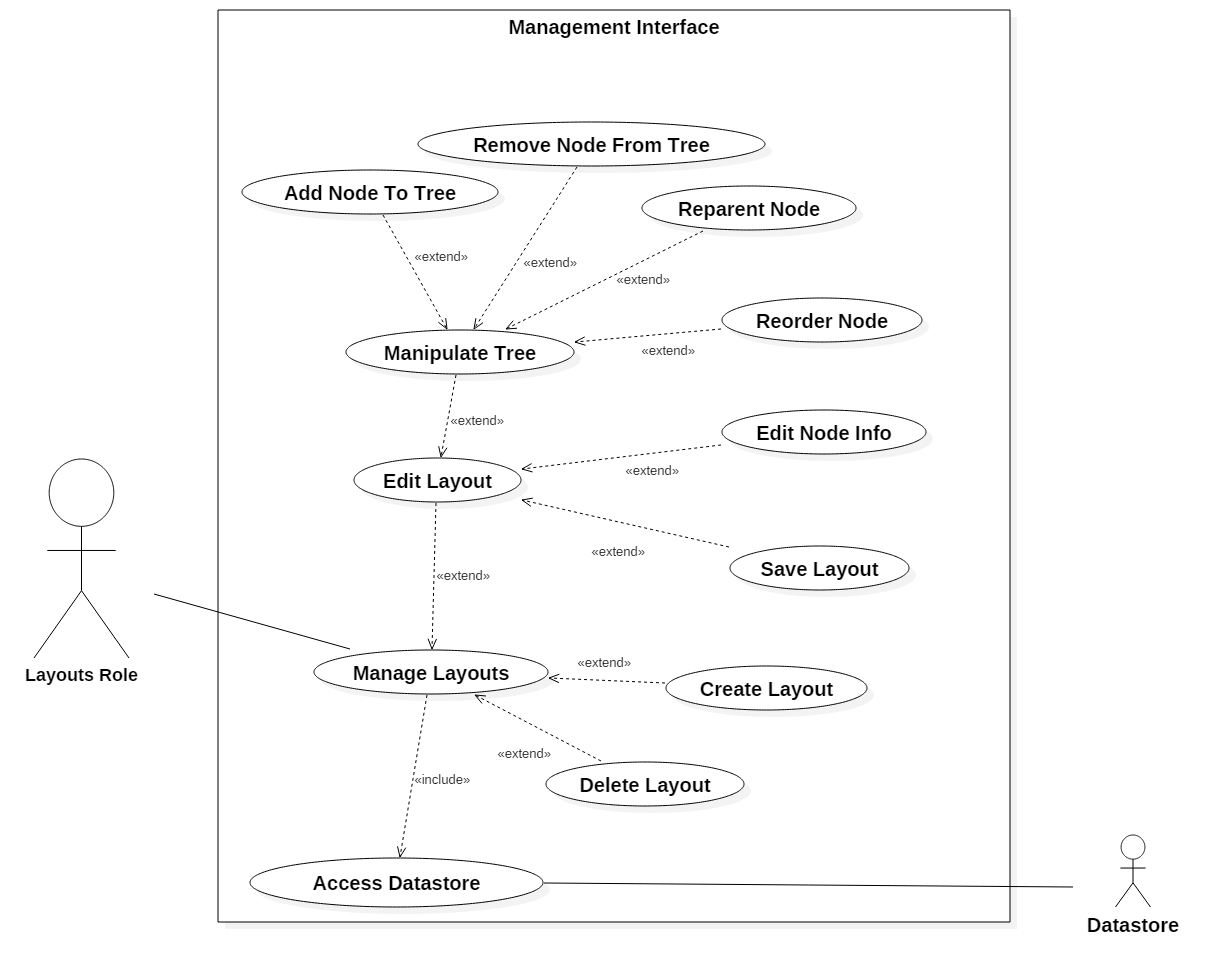
**Player File System**: The file system on the player device, receiving metadata and media files for rendering.



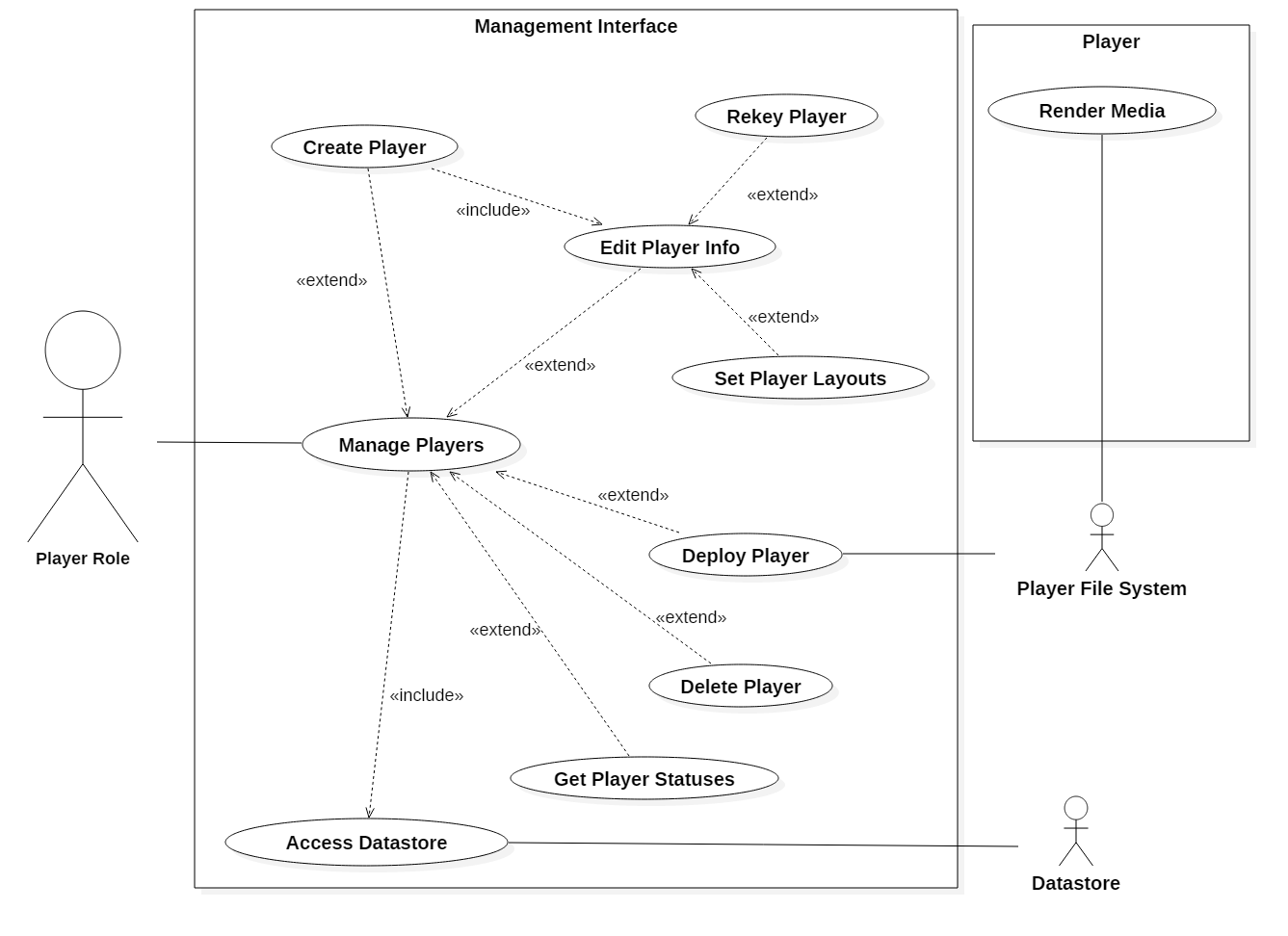
*Figure 4.3-1: Use Case Diagram - General use and User Manager*

## 

*Figure 4.3-2: Use Case Diagram - Media Management*

**

*Figure 4.3-3: Use Case Diagram - Layout Manipulation*

**

*Figure 4.3-4: Use Case Diagram - Player Management*

### 

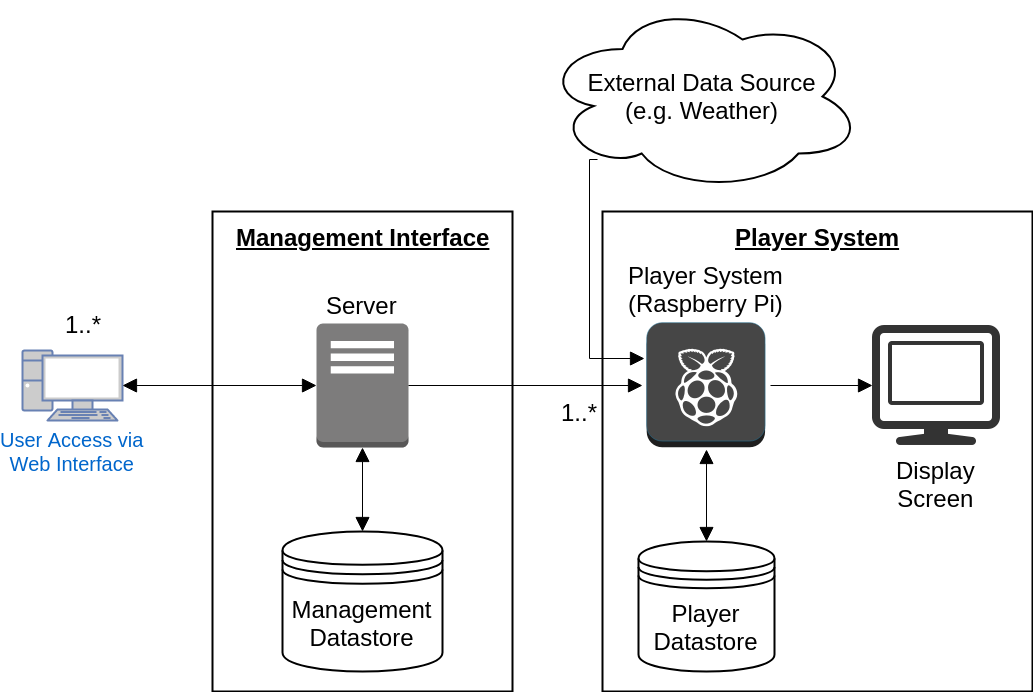
### 

### 

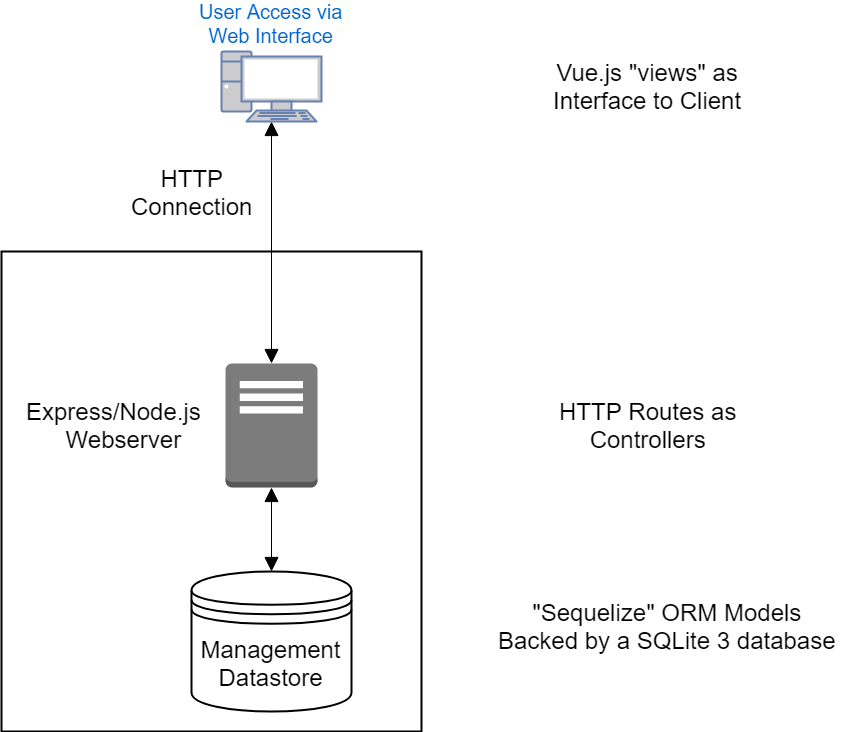
# **5. Software Architecture**

## ***5.1. Overview***

The overall architecture of the inspiring solution (Four Winds) was well fitted to our task, with some minor modifications. Clients will interact with a remote server hosting the management interface (previously a local application connected to a remote database). The management interface will deploy data to individual signage players, which will render the content to attached display devices. After a push, individual players will have no need to interact with the management interface, but may contact third party servers for certain data updates (e.g. weather information) if so configured by a user in a layout.



*Figure 5.1-1: Signage system architecture*

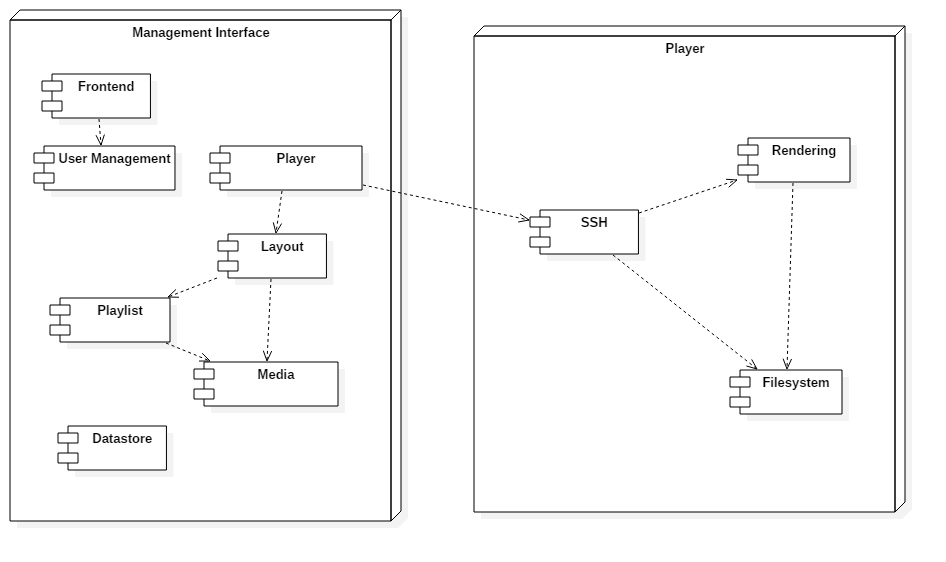
**

*Figure 5.1-2: Model-View-Controller layers in the Management Interface*

## 

## ***5.2. Subsystem Decomposition***

The following is a high level overview of the components of the overall system, including those provided by the environment and 3rd party software.



| **Frontend** | HTTP/HTML interface facilitating user interaction |
| --- | --- |
| **User Management** | User authentication and permissions |
| **Player** | Handles connection and deployment to the player system |
| **Layout** | Handles layout tree containing media/playlist for display |
| **Playlist** | Contains multiple media items for displaying in sequence in one location |
| **Media** | Handles individual media items |
| **Datastore** | Database and file store for objects on the management server |
| **SSH** | Allows for media and metadata deployment (SCP) in addition to signalling (e.g. to restart the renderer) |
| **Rendering** | Displays the deployed media, chromium based system |
| **Filesystem** | Standard Linux filesystem on the Raspberry Pi Player |

## ***5.***3***. Persistent Data Management***

### Management Interface

The management interface contains both a SQLite 3 database as well as copies of uploaded media files on the local filesystem. Access to the database will be facilitated through the “Sequelize” ORM (object relational mapping) for Node.js. This layer allows for access to object without the direct use of SQL. Media files will be stored with the operating system provided file system.

### Player System

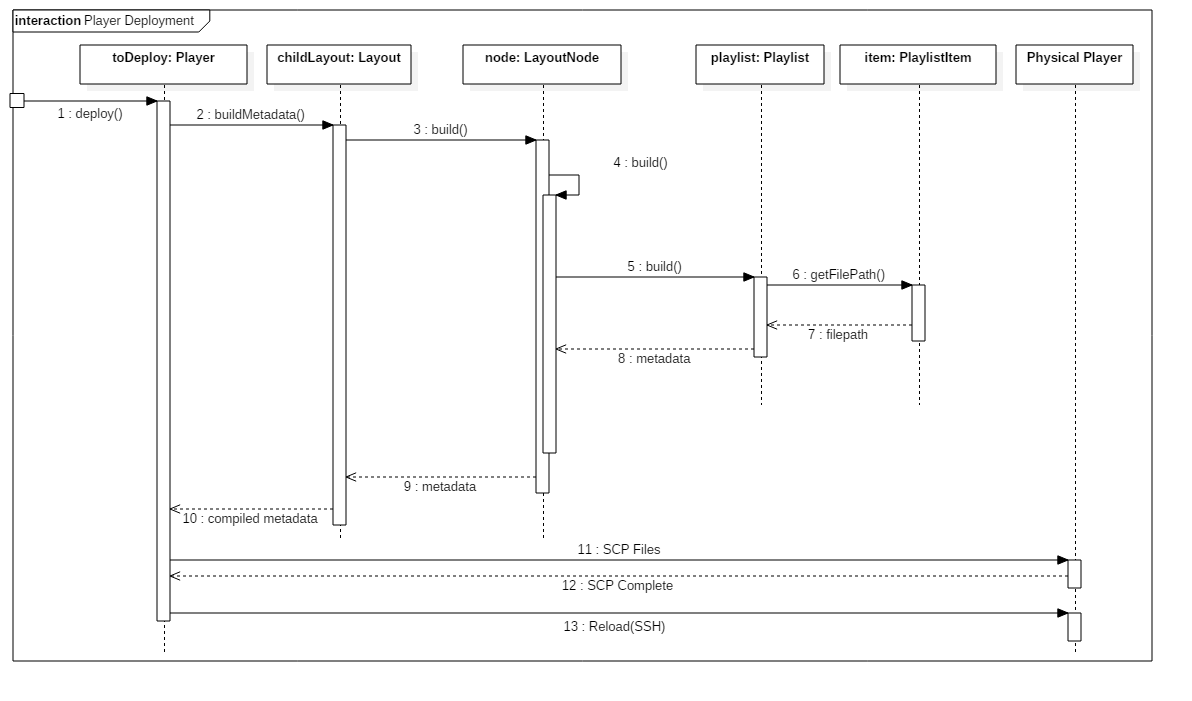
The player system will store both media files and metadata necessary for rendering on the operating system provided file system. Metadata is currently composed of HTML files representing the layout and properties of the rendered content.

# 6. Object Design

## ***6.1. Overview***

This chapter will build on the architecture and design. We examine some important interactions within the system and detail the class relations in the system.

## ***6.2. Object Interaction***



*Figure 6.2-1: Sequence Diagram for deploying a player*

The process for building and deploying a sign to the physical player involves two major steps. First the data necessary for rendering must be compiled: HTML layout file and media files. This process must be performed recursively on the layout tree, each node returning an HTML fragment and file listing for itself and its children. Secondly, the gathered data must be transferred to the physical player and a signal must be sent to inform the device to reload.

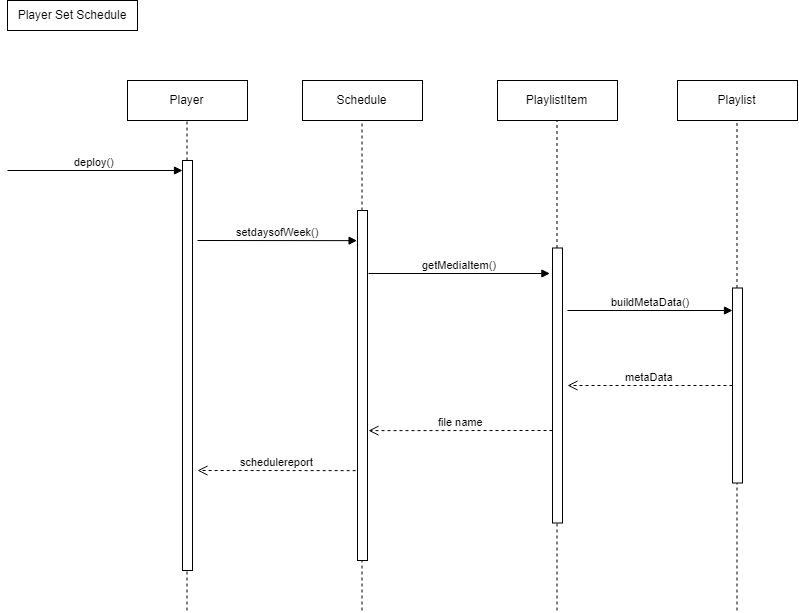
**

Figure 3-1: Sequence Diagram for schedule change

## ***6.3. Detailed Class Design***

The following is a class diagram representing all persisted classes (saved to database) in the management interface system.

# 

The user related classes (highlighted in blue), facilitate security in the management interface. Each user of the system has an associated User object in the database, with their associated roles. Whenever a user logs in from a new computer, a Session is created, tying their browser session to their User object.

The media subsystem (in red) consists of image files uploaded as a *MediaItems*, and Media Items are located in a *MediaItemFolders*. This forms a tree analogous to a filesystem, with folders containing subfolders and items, thus providing a method of organization to the users.

The Layout class represents a configuration of a screen, detailing where certain items appear. Each layout exists as a tree of *LayoutNodes* (in green). A *LayoutNode* can be organizational (e.g. dividing the screen into sections) or functional (e.g. showing the time). Playlists are a special type of LayoutNode, containing several PlaylistItems which refer to MediaItems, showing them on the screen in sequence. Each *LayoutNode* may have several dynamic properties specified in *LayoutNodeProperty* objects, based on its specific type.

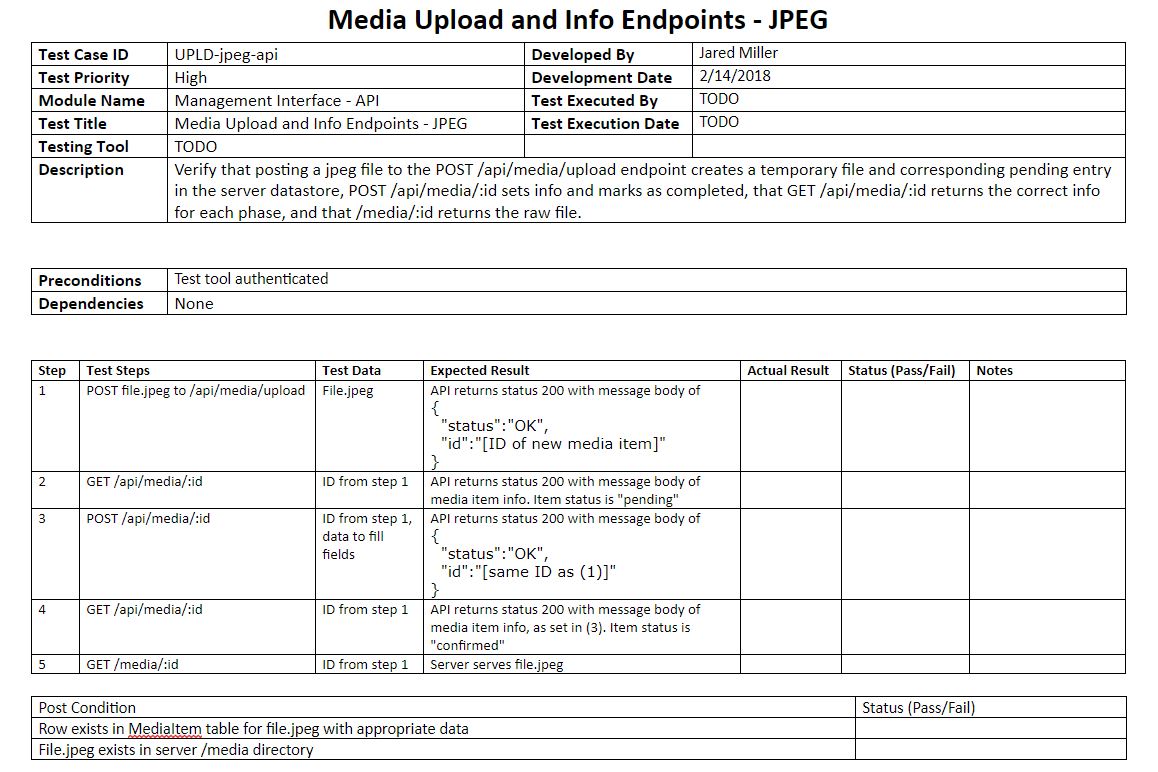
# **7. Testing Process**

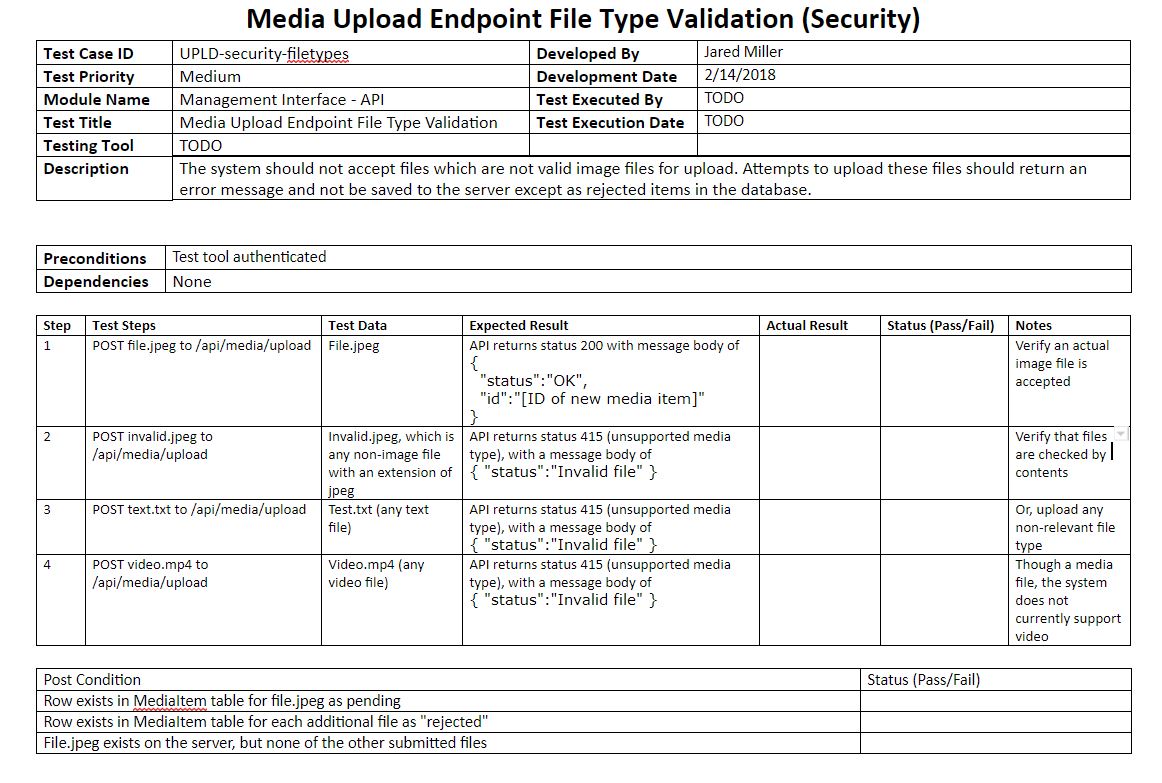
7.1. User Experience Tests

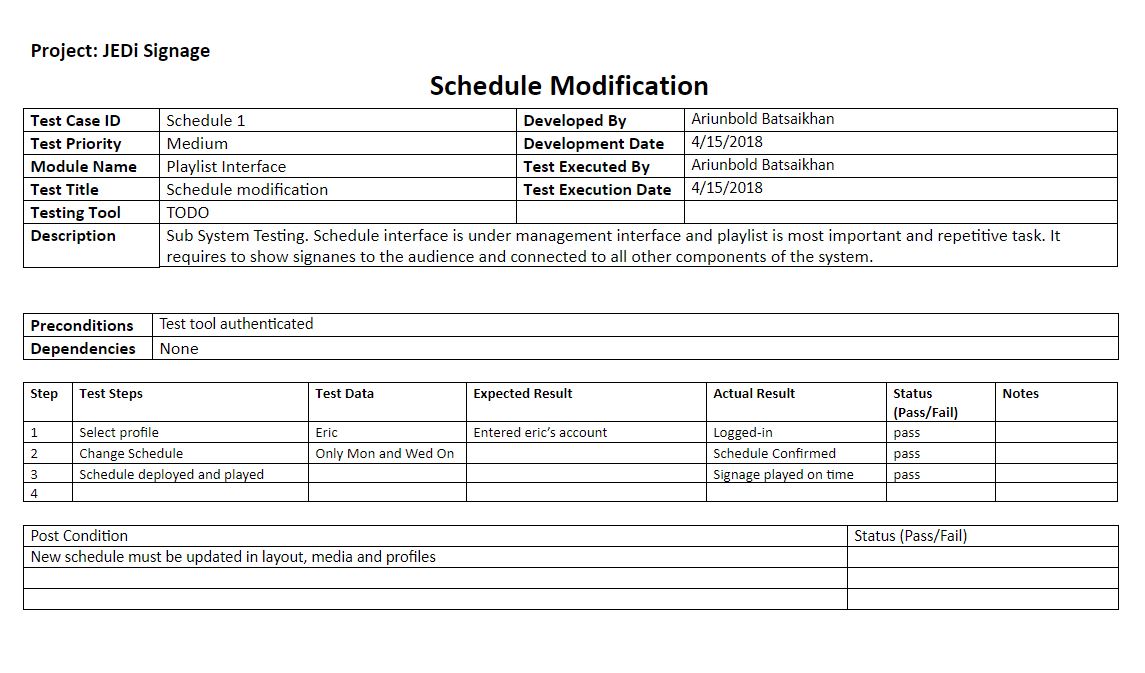
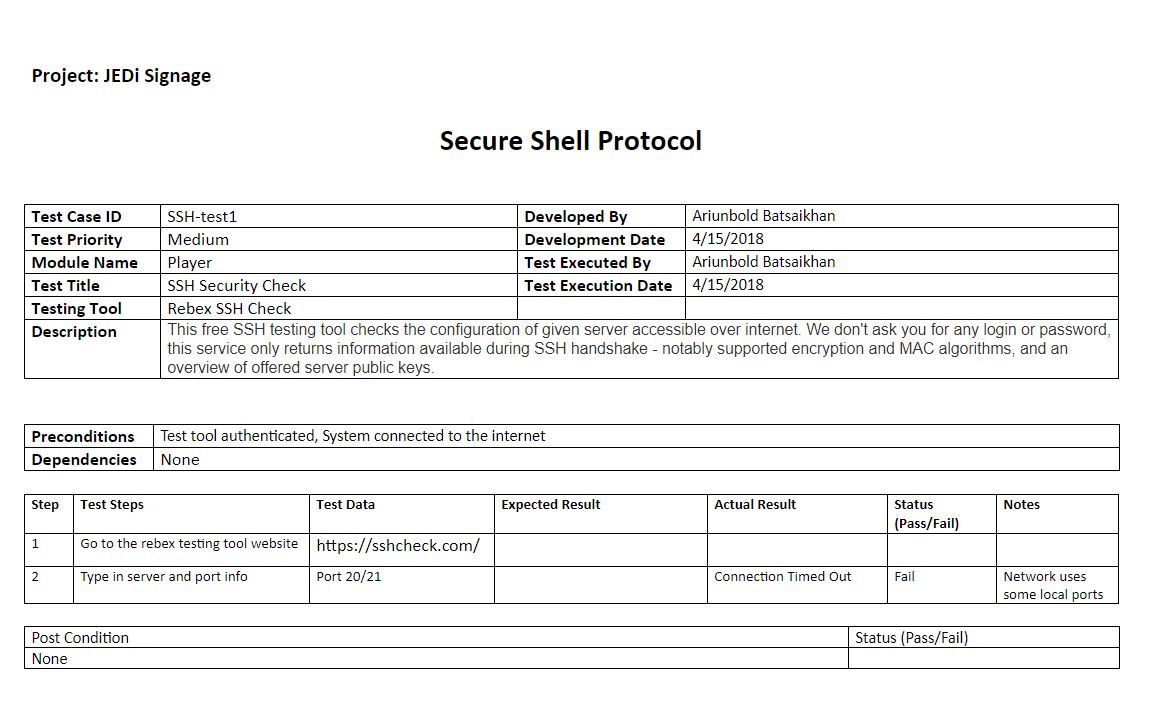
James is a student and is part of the cheese club and wants to upload an event flyer for his cheese event. James attempted to use the system to upload an event flyer to the tv screen in the University Center. He didn’t find the system hard to use it was relatively easy for him to upload his cheese event flyer. He was able to figure out how to keep his flyer up on the screen for a week prior to his cheese event date.

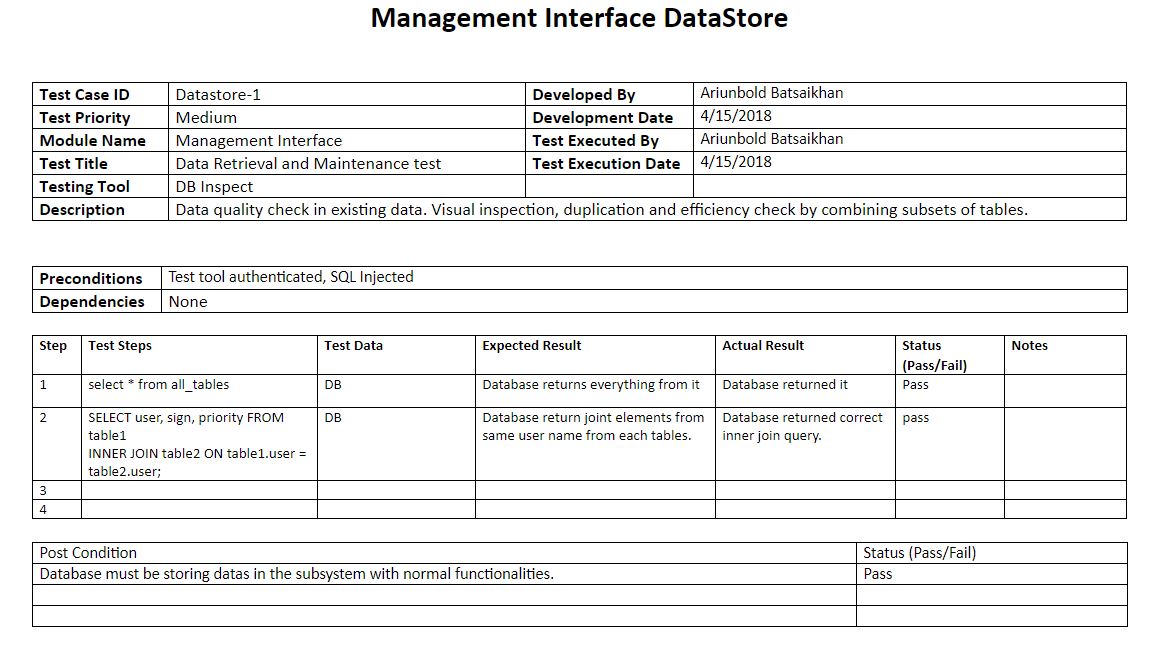
Sam is a psychology professor and is doing a research study that requires student involvement. Sam wants to upload the flyer to participate in her research study on the tv in the skywalk from french hall to the library and the tv in the MSB breezeway. Sam didn’t find it hard to upload her flyer to both tv’s. Sam thought it was a little bit of lag time uploading but it wasn’t too long of a wait.

7.2. Systems Tests





7.3 Subsystems Tests



# 

# **8. Glossary**

# 

| Datastore | For this project, a combination of a database (SQLite) and a file store. |
| --- | --- |
| Express | Node.js library used to manage an HTTP server, routing, and other middleware |
| HTML | HyperText Markup Language. Defines the appearance and functionality of the user interface. Also used in this project to define a layout to a Player. |
| Layout | A set of configuration specifying how content is displayed on a player. |
| Media, Media Item | An image file |
| Middleware | Functionality that manipulates an HTTP request in some way (e.g. checking for cookies, handling an upload, setting headers) |
| Node.js | A JavaScript (or ECMAScript) runtime based on the chrome V8 JavaScript engine. |
| Player | A “content player”, responsible for displaying media and information to the user through a screen. For this project it is contained on a Raspberry Pi. |
| Role | A set of permissions assigned to a user. |
| Sequelize | An ORM (object relational mapping) library used to define, access, and manipulate objects in the system. For this project it is configured to use a SQLite database. |
| SQLite | A relational database system which stores data in a single, portable file. |
| SSH | Secure Shell. Used to transfer data and status information between the management interface server and the players. |
| Vue.js | “The Progressive JavaScript Framework.” Used in the management interface to provide dynamic HTML rendering, interactivity, and server communications. |

# 

# **9. Appendix**

## 

## 

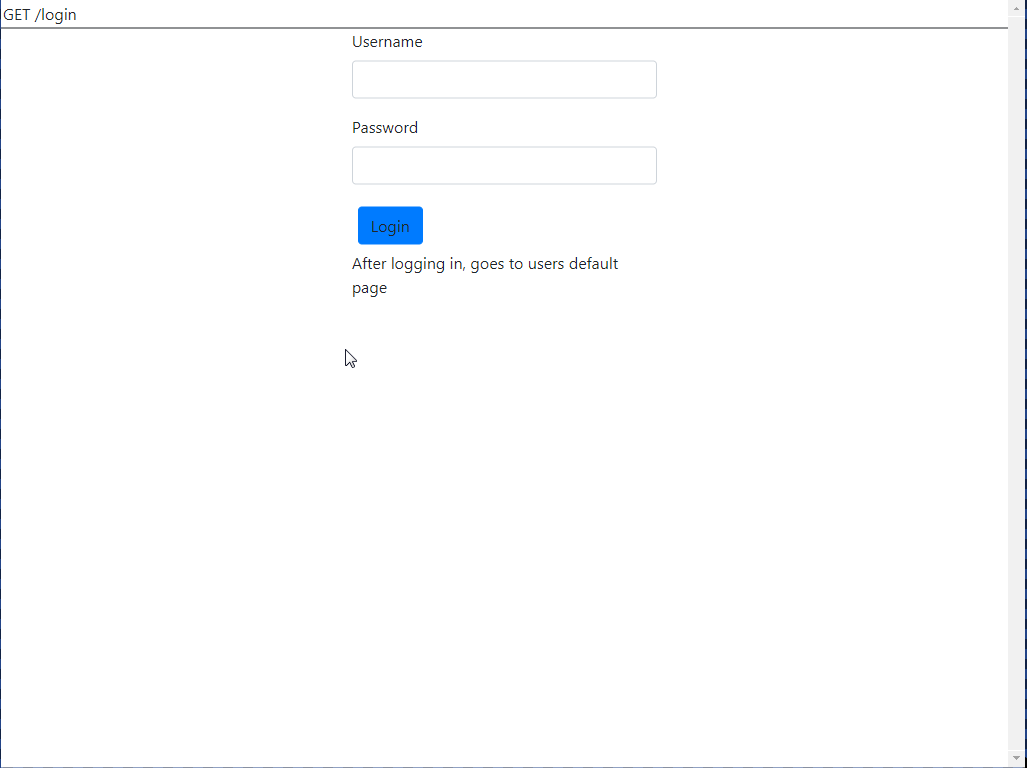
## 

## 9.1. Appendix A - Use Cases

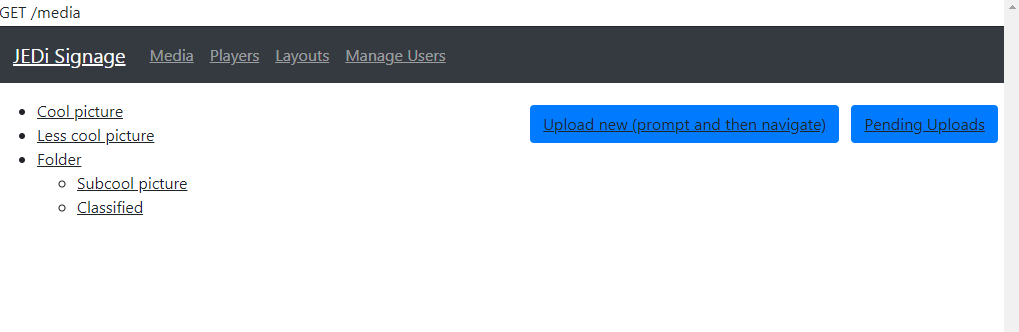
See the following Pages

## ***9.***1***. Appendix*** B ***– User Interface Designs***

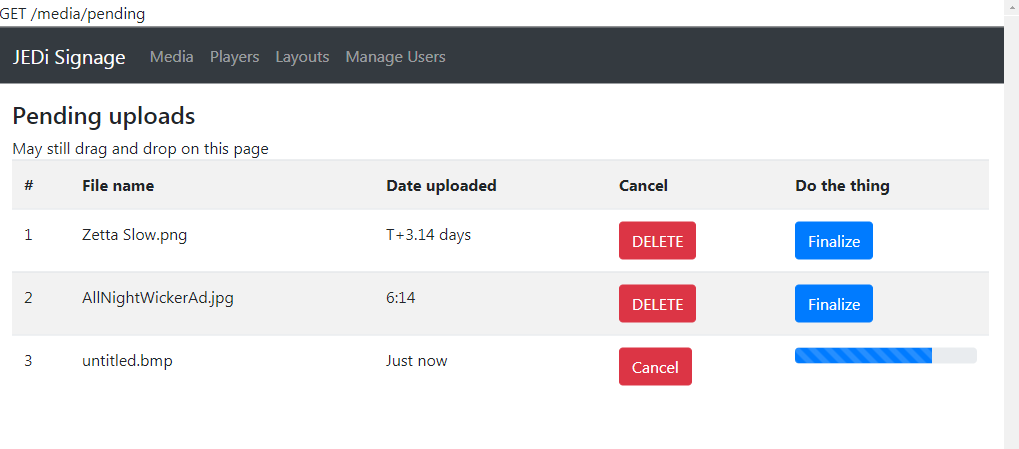
The following are screenshots from the storyboard created in the early design phase.



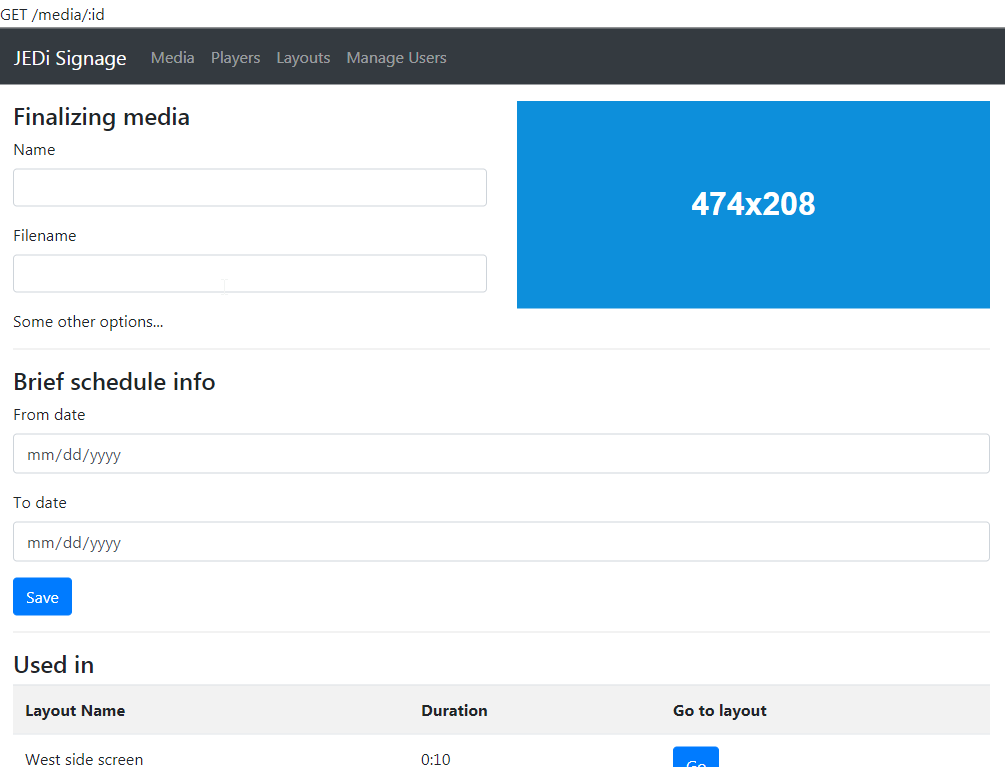
*Figure B-1: Login Screen*

**

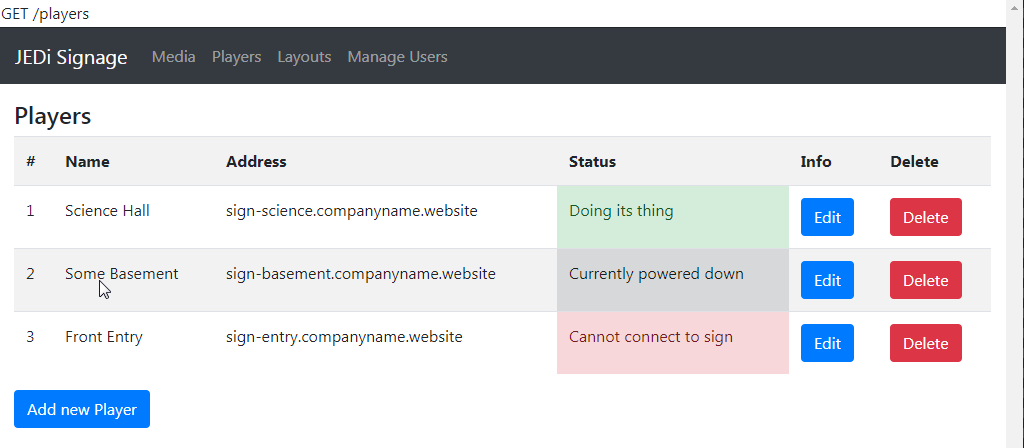
*Figure B-2: Media listing screen*

**

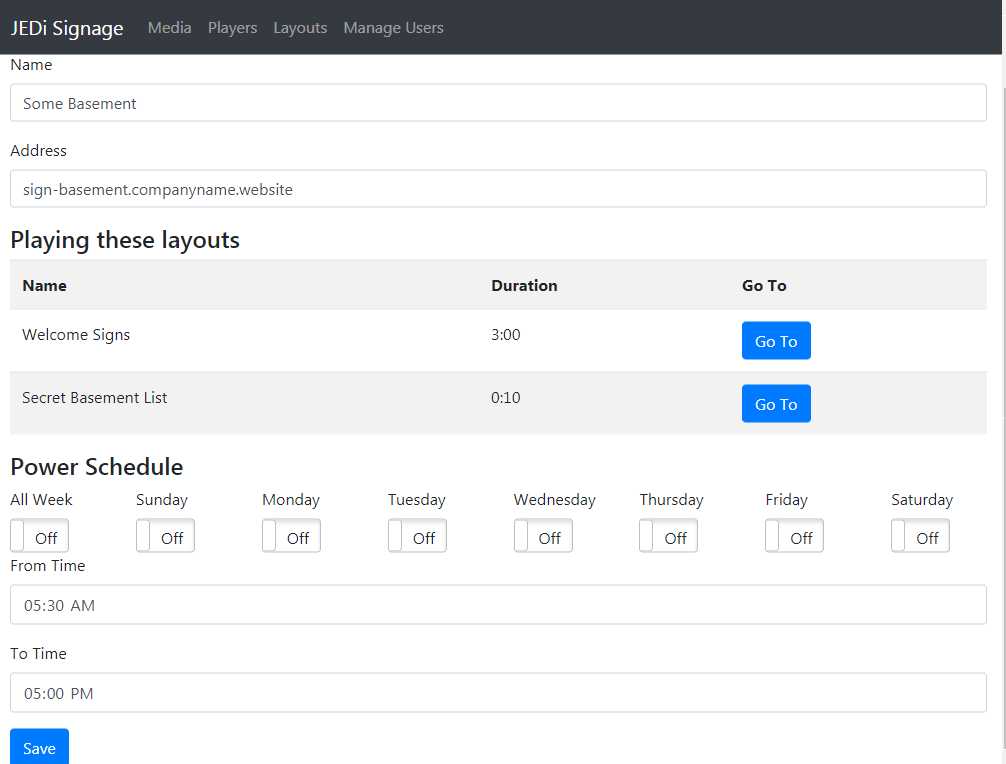
*Figure B-3: Media Upload Screen*

**

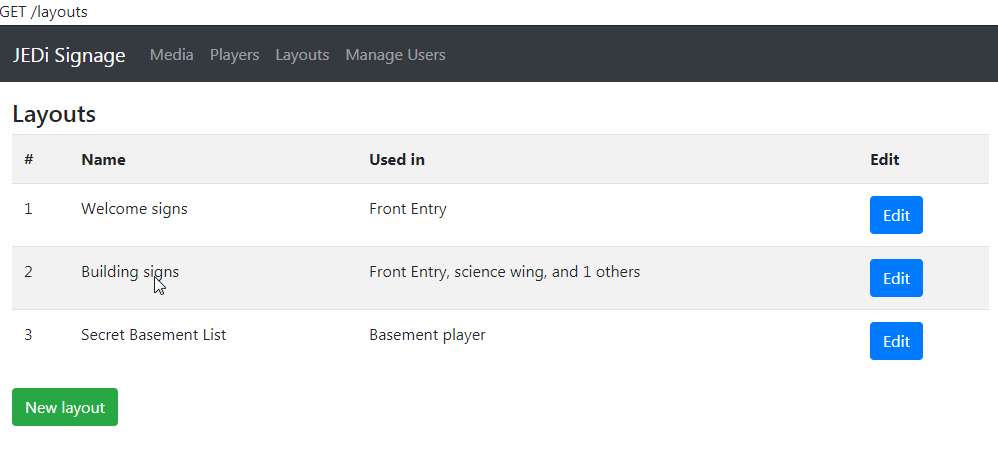
*Figure B-4: Media Info Screen*

**

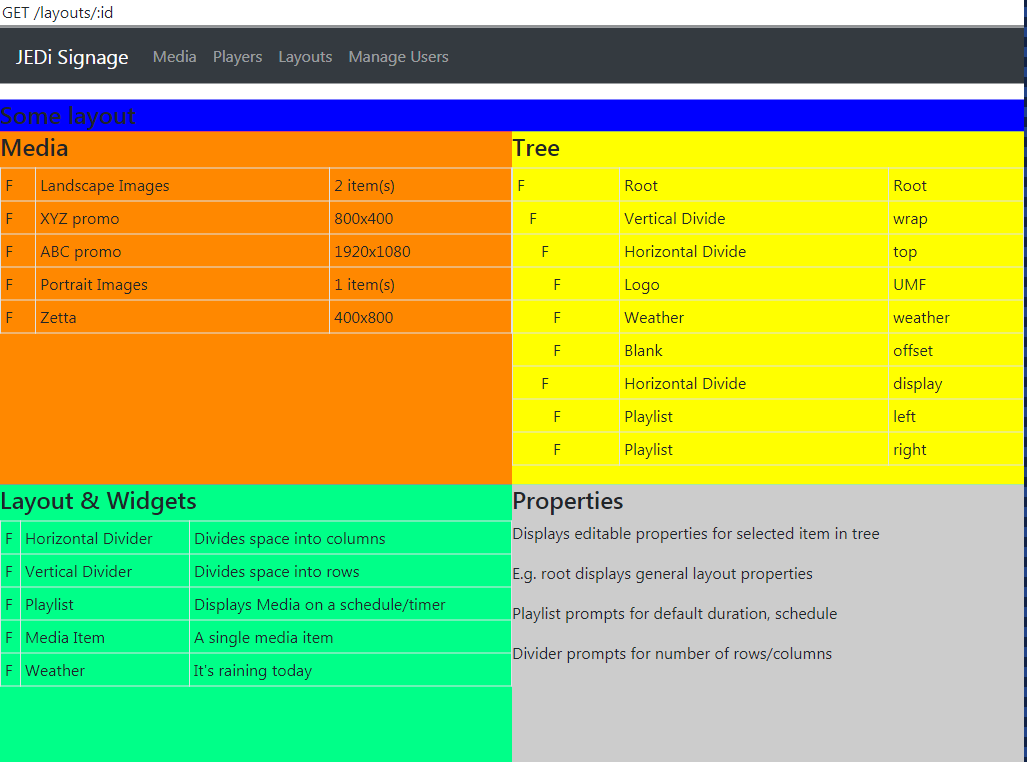
*Figure B-5: Player Listing Screen*

**

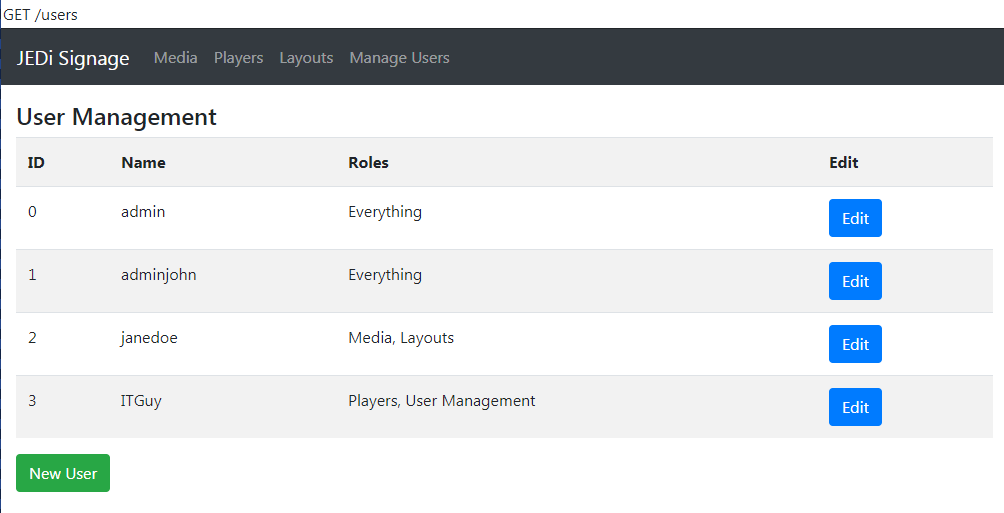
*Figure B-6: Player Info Screen*

**

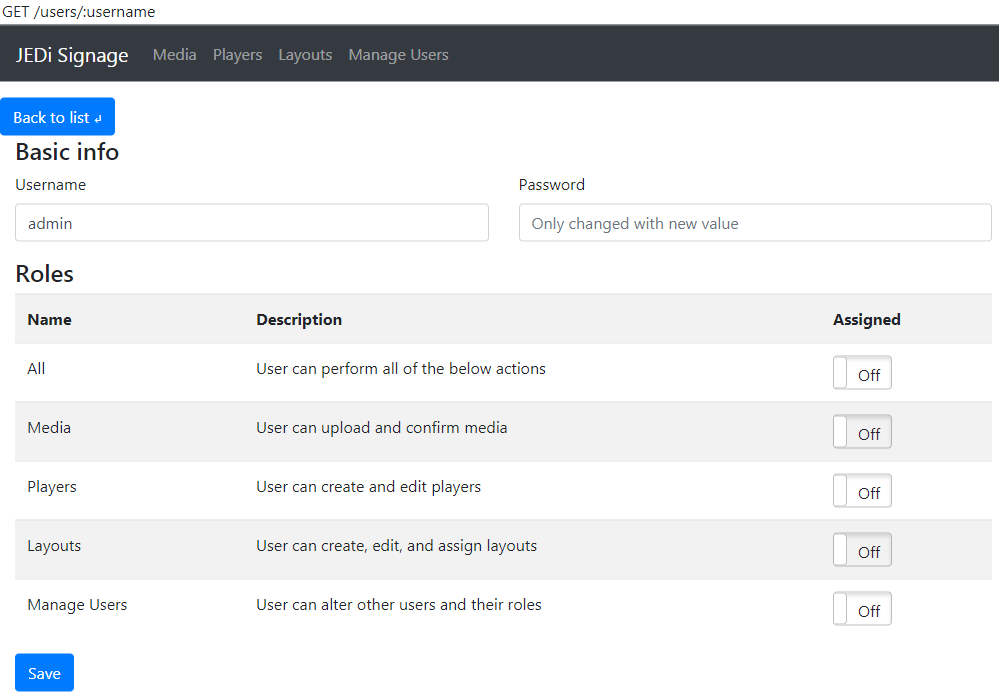
*Player B-7: Layout listing screen*

**

*Figure B-8: Layout Editor - early design*

**

*Figure B-9: User Listing Screen*

**

*Figure B-10: User Editing Screen*