SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

Date: December 3, 2016

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Kory Bartlett
Farzad Fred Feyzi

ENTITLED

Shot Clock

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

_____
Thesis Advisor

_____
Department Chair

Shot Clock

by

Kory Bartlett
Farzad Fred Feyzi

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
December 3, 2016

Shot Clock

Kory Bartlett
Farzad Fred Feyzi

Department of Computer Engineering
Santa Clara University
December 3, 2016

ABSTRACT

The sports sector has experienced major growth over the past couple of decades and is anticipated to grow even faster in the foreseeable future, especially as media technology continues to advance and access to sports games continues to increase. As the audience of sports enthusiast and observers expands, more people may miss the most recent and interesting sporting events from the previous night. Many websites have attempted to serve these groups, but they have become cluttered with articles or short-clips that hold little to no significance to the sporting outcome. These websites have begun to get away from the simple need of highlighting an event in an efficient format. Our solution, Shot Clock, is a simple, centralized web application to view sports highlights that functions by taking advantage of today's modern technologies. The web application will also provide a user experience that promotes simplicity, continuity, and ease of use when navigating the application. Users will also have the opportunity to create custom lists so that they can be kept up to date to their favorite teams. Succeeding in these major facets will provide audiences with the practicality of easy access to precise replays delivering an innovative experience that users will appreciate.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The sports industry in North America alone currently is worth 60.5 billion dollars and this industry has been projected to grow to 73.5 billion dollars by 2019. This rise in projected growth happens due to the fact that media companies are making larger deals, which in turn is caused by more people being entertained by sports. However, not everyone can make it to the television for every game, and not everyone has the time to watch a full-length game. This creates the need for easy access to fast sports highlights and other sports statistics that can help individuals relive the exciting moments in a match/game without having to spend too much of their time watching the actual event. Therefore, a user should be able to navigate to a website and within a few seconds be watching the game he or she intended to watch. Quite a few of these websites currently exist today, but we think that they could be dramatically improved.

There are many websites today that have attempted to solve this issue, but while attempting to cover all sports related issues, they have gotten away from the simple goal of highlighting sports in a precise fashion. Websites like ESPN.com, FoxSports.com, and BleacherReport.com all encompass some of the highlights from a particular sports league, but they all have become cluttered with loosely related sports articles and irrelevant game interviews. The current methods include too many articles, and these articles often take up too much space within a users window. This is troublesome because the user should simply get to the video highlights in a fast manner, without having to traverse the entire cluttered site for over 30 seconds; the search time must be decreased. In addition to that, when it comes to different sports, the current implementation does not have any continuity between the sports; in other words, the view for soccer typically varies significantly from the view for basketball. Website usability is key for user experience because discontinuity between key pages can lead to frustrations causing a user to discontinue use of a website. Lastly, many current implementations do not have a reliable source of efficient highlights. For example, some websites will leave out certain games within a league, creating an inconvenience for the user who is just trying to view his or her

desired highlight.

1.2 Solution

Our solution is to use modern technologies in order to create a simple, centralized web application in which users can easily view any highlight within any sport and league. We will make sure there is continuity between different sports pages by using a one-size fit all view. In addition to that, there will be unique customization features, so highlights can be catered to users interests. The system will pull videos and gifs with thorough content ensuring fans can experience a full game in a matter of minutes, and we can will guarantee no game within a season is missing. The window for viewing video will be conveniently placed in the middle of the website, making it easier for people to see, with navigation bars around that centralized feature. We will not include articles because that takes away from the viewing experience and makes it harder to search for highlights in a timely manner. Overall, the different, more simple user experience along with more unique customization and guaranteed highlights will provide users who are interested in highlights the ideal platform for them to follow their teams and games.

Chapter 2

Requirements

2.1 Introduction

Requirements define and qualify what the system needs to do. Functional requirements specifically define what must be done and what the solution must do. Non-functional requirements have to do more with performance and reliability and thus, talk about how the functional requirements need to be achieved. Lastly, design constraints constrain the solution rather than the problem.

2.2 Functional Requirement

- Critical
  - The system will allow the user to search for videos
  - The system will allow the user to see a steady stream of videos
  - The system will allow the user to add videos to his video queue
- Recommended
  - The system will give the user the ability to sign in and sign out
  - The system will allow the user to customize his or her settings in order to create a more personalized web app
  - The system will not allow for repeat username

2.3 Non-Functional Requirement

- Critical
  - The system will be put together using user friendly UI

  - The system will be easy to navigate
  - The system video portal will be consistently laid out between pages
  - The system will be extensible to work for varying sports leagues

- Recommended
  - The system will be secure and have a well tested sign in process
  - The system will be responsive

2.4 Design Constraints

- Critical
  - The system must be web based

- Suggested
  - CSS frameworks and other modern libraries should be used because of its ability to create good looking UI
  - Sport themed images and icons are encouraged in order to go with the theme of the project

Chapter 3

Use Cases

3.1 Introduction

This section describes the four to five main use cases that a student can perform. The first subsection contains a diagram displaying the main user and the user's four to five possible actions. The second subsection details the name, goal, conditions, and steps corresponding to each use case.

3.2 Diagrams

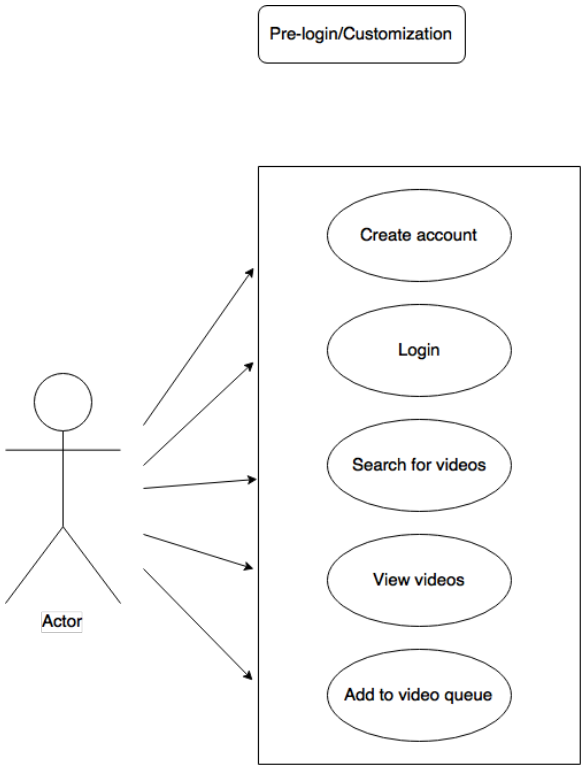

Figure 3.1: Use Case Diagram of Pre-login

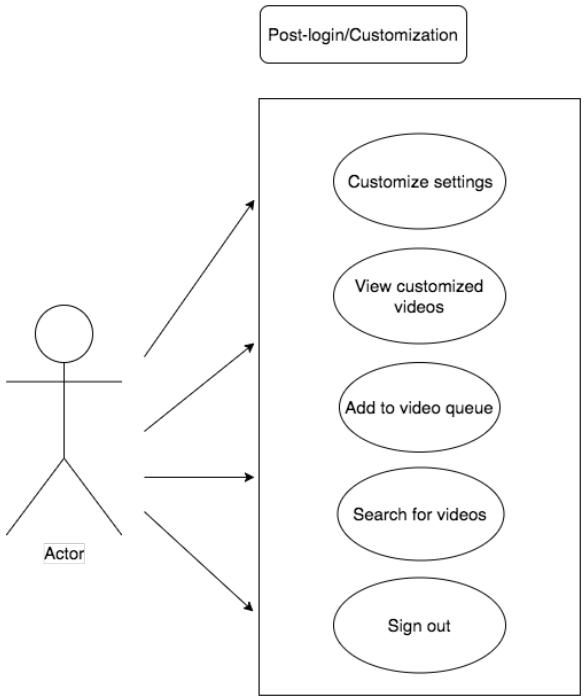

Figure 3.2: Use Case Diagram of Post-login

3.3 Use Case Description

- Create account
  - Name: Create an account to be used on website
  - Goal: To create an account ti be used on website
  - Actor: User
  - Preconditions
    1. User must be on webpage
  - Steps
    1. User selects my settings
    2. User selects create an account
    3. User enters desired username and password combination
    4. User submits username and login combination to system to be submitted
  - Post-conditions
    1. Assuming user's username is unique, the combination gets committed to the system
    2. The website loads user's stored data and allows for user customization

3. The website returns to the overall view of the website with user now logged-in

- Exceptions

1. User's username needs to be unique so process may need to be repeated to achieve a unique result, user will be alerted of error occurring

- Login
  - Name: Login to website
  - Goal: To sign in to website
  - Actor: User
  - Preconditions
    1. User must be on webpage
    2. User must have a preexisting account
  - Steps
    1. User selects my settings
    2. User can create an account to be used to login
    3. User enters login credentials
    4. User submits login to be signed into website
  - Post-conditions
    1. User's username is checked if it exists in the system
    2. The matching username and password are returned
    3. The user's password is then checked against the returned username's password
    4. Assuming the credentials match, the user is then logged on to the system
  - Exceptions
    1. User who has entered information but has not created an account will be asked to create an account
- Search for Videos
  - Name: Search for videos
  - Goal: Allows user to specify videos to watch
  - Actor: User

  - Preconditions
    1. User must be entered into system
  - Steps
    1. User types into textbox to specify which sports video to view
    2. User video returned ready to view
  - Post-conditions
    1. User continues on website to view video
  - Exceptions
    1. Video requested by the user is unavailable in database of video sources, so user is alerted of unavailability of video and can re-enter search items to perform another search

- View Videos
  - Name: View videos
  - Goal: To watch video on website
  - Actor: User
  - Preconditions
    1. User must be entered into system
    2. User must allow video to play
  - Steps
    1. User able to enjoy video playback
  - Post-conditions N/a
  - Exceptions N/A

- Add to Video Queue
  - Name: Add videos to queue
  - Goal: To use website to add user specific videos to queue
  - Actor: User
  - Preconditions
    1. User must be entered into system

2. User must have already searched for videos

- Steps
  1. User types into textbox to specify which sports video to view
  2. Video is added to the back of the queue
  3. Video is ready to be viewed in queue order
- Post-conditions
  1. To view video user must allow video to play through
- Exceptions
  1. If the user has cleared their queue of videos after the new video is added to the queue then the recently added video along with other videos queued for viewing will be removed from queue. System will play the newest video added next

- Customize Settings
  - Name: Customize settings
  - Goal: To customize personal settings
  - Actor: User
  - Preconditions
    1. User must be entered into system
    2. User must have an account and be logged into system
  - Steps
    1. User clicks my settings tab
    2. User can edit their username and password
  - Post-conditions
    1. User must save changes made when editing their profile
  - Exceptions
  - *unable to udpate*

- View Customized Videos
  - Name: View customized videos
  - Goal: To view list of customized videos referred to as MyReplay

  - Actor: User
  - Preconditions
    1. User must be entered into system
    2. User must be logged into system
    3. User must have edited and saved their MyReplay preferences
  - Steps
    1. User returns back to home page of website
    2. User can enjoy video playback of queue
  - Post-conditions
    1. The program grabs the desired data entries
    2. Videos will play in most recent found from database
  - Exceptions
    1. User may watch an old video if the update has not been run by program

- Sign Out
  - Name: Sign out of website
  - Goal: To sign out the users profile from the website
  - Actor: User
  - Preconditions
    1. User must be entered into system
    2. User must be signed in to the system
  - Steps
    1. User clicks my settings tab
    2. User clicks log off button
    3. User profile has exited the system
  - Post-conditions
    1. Clear user data from the website
    2. Reset the website so it functions as if a new user has just entered
  - Exceptions N/A

Chapter 4

Activity Diagram

4.1 Introduction

This activity diagram outlines the basic flow of user activity while using the our webpage. Users, upon entering the site, can choose to sign-in to their accounts for customization or continue to the site without features of customization. The user can then search for specific videos to view. If the user decides to not search for a specific video then they will view the queue of videos available.

4.2 Diagram

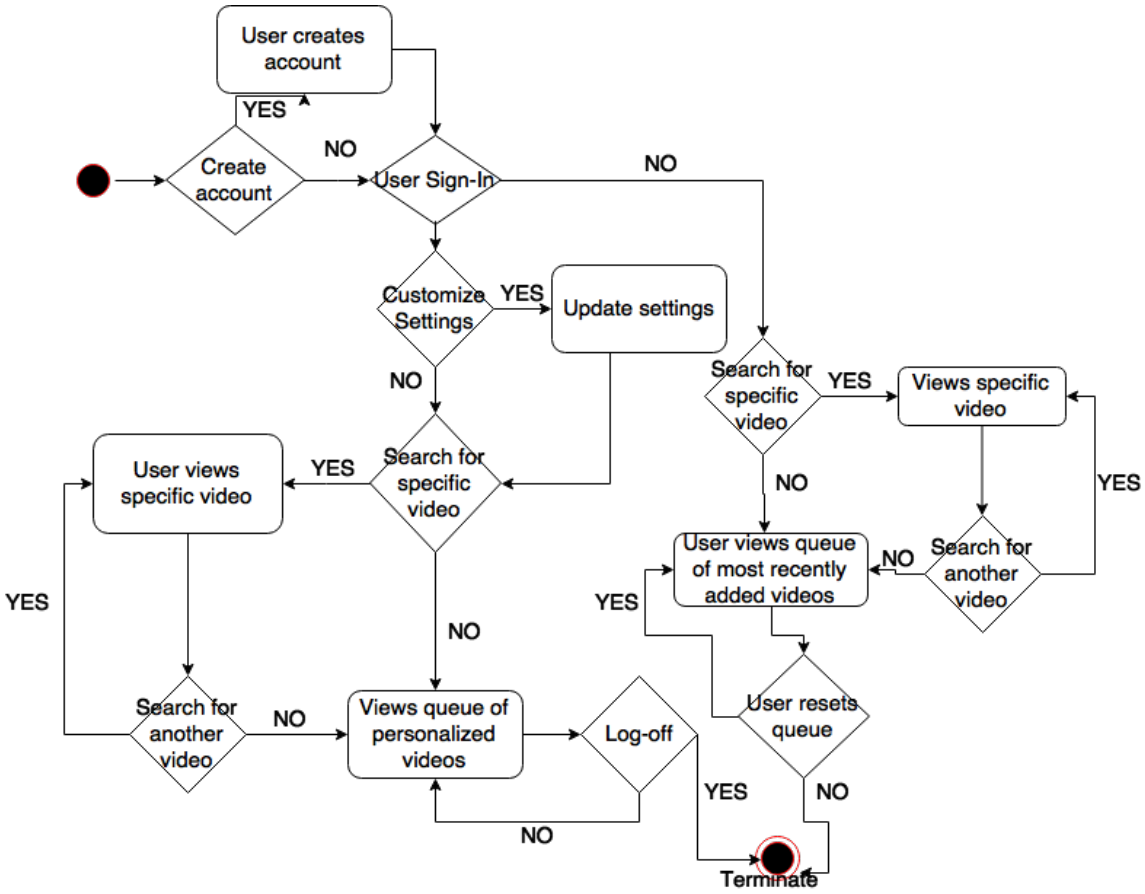

Figure 4.1: Architectural Diagram

Chapter 5

Mock ups

5.1 Introduction

The conceptual model illustrates through mock ups how the system will appear to different actors who might use the system. The webpage will look the same overall, however it has small differences between users who decide to create an account and users who dont have an account. Figure 5.1 shows the overall view. This remains the same for both users. Figure 5.2 shows the settings that can be manipulated under My Settings. Figure 5.3 shows the page for how one would make an account.

5.2 Mock Up Diagrams

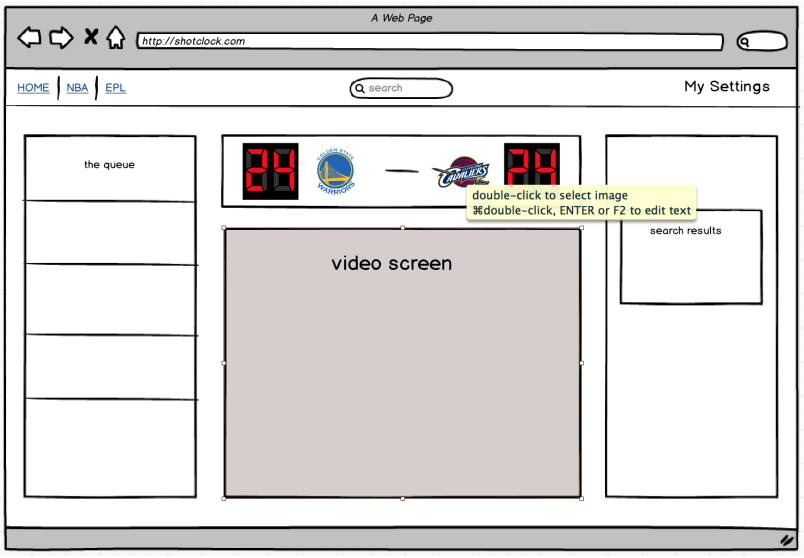

Figure 5.1: Overall view of website

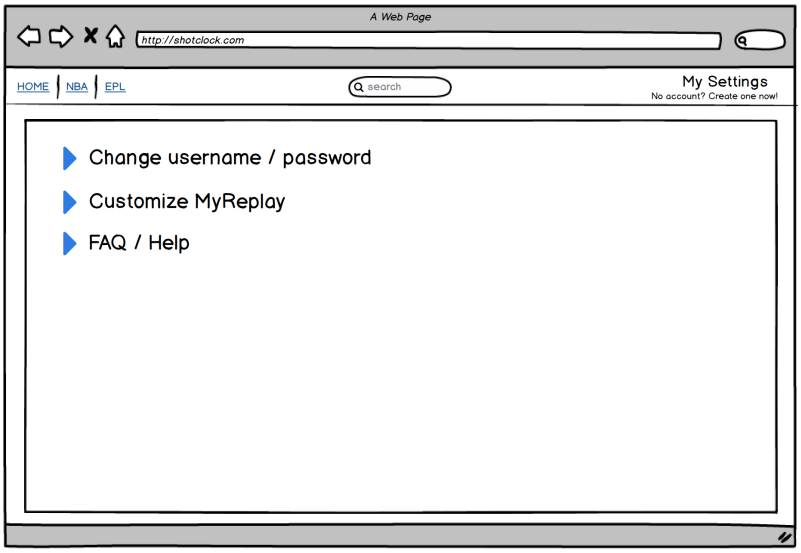

Figure 5.2: My settings

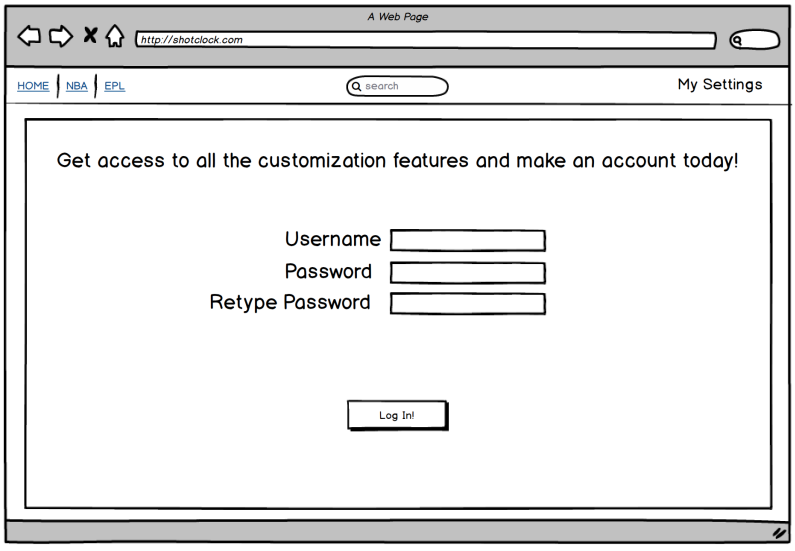

Figure 5.3: Account creation

Chapter 6

Technologies Used

The technologies used outlines the current technologies and languages used to complete the project. These technologies were chosen because of their ability to provide the services needed to complete the requirements. These technologies are listed below:

- HTML - Front end language used to serve as the structure for the website solution
- CSS - Front end language to help style the website solution
- Javascript - Programming language to be used for both front and back end
- Angular JS - Two-way data binding for responsiveness and part of the MEAN stack.
- Ajax - Added to help make data population and page loading asynchronous
- Node JS - The back end language that will be used to help connect with our ElasticSearch instance and will be the running server to handle the web crawler and event handlers
- ElasticSearch - An open-source, easily scalable search engine based on Lucene that serves to hold and index data
- Materialize CSS - A front end library provided by Google for styling purposes
- Express JS - A specific framework within Node that helps out with route handling and middleware creations
- MongoDB - Another part of the MEAN stack that will be used for creation of the models in our application

Chapter 7

Architectural Diagram

7.1 Introduction

The Architecture Diagram illustrates the kind of software architecture used to create the system. An architecture allows the way the system functions to be easily understood. Our applications system uses a client-server architecture, illustrated in the figure below.

7.2 Diagram

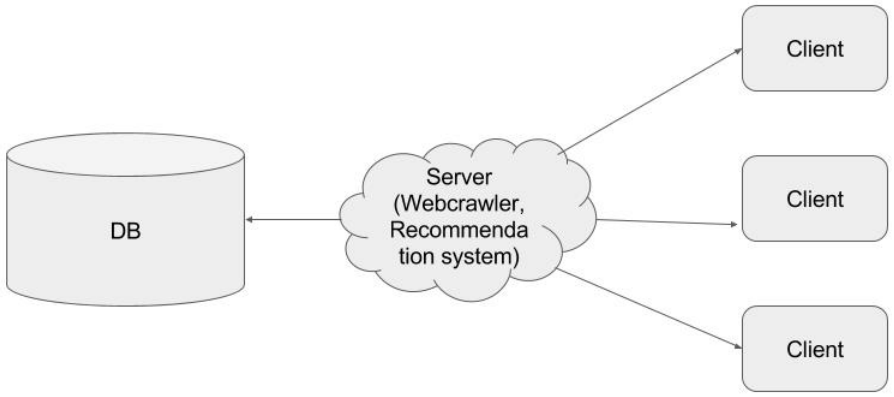

Figure 7.1: Architectural Diagram

7.3 Architectural Rationale

As stated above, our solution uses the client-server architecture. Users, or clients, can access the application through its url. The server then does a lot of the event handling. The server is responsible for having a running web crawler on a cron job that will seek to obtain data, and then index the data appropriately in our real time ElasticSearch database. The server will also be handling the recommendation system as well, so users can have the data theyre customized data. The server will then connect to the database to store the links to the videos (so we dont need to actually store massive files). The database will have easily indexable data.

Chapter 8

Design Rationale

8.1 Introduction

The design rationale gives reasons for the decisions we made in the design of the product. This primarily justifies the choices made about the user interface design and the technologies used.

8.2 User Interface Design

The user interface for this application will have the look and feel of a single page application (SPA), but will consist of numerous pages; thus, requiring numerous routes. We chose this because we want to account for continuity among pages, but also prioritizing the viewing experience. Except for having to make changes to his or her account, the user should always be able to have a running video feed on the application. Having an interface as such will keep the user busy watching, while being able to do numerous other tasks (like searching for videos) at the same time.

8.3 Technologies Used

The technologies used are basic, yet modern web technologies and frameworks that should render similar pages across multiple browsers. We will use an ElasticSearch instance along with a MongoDB database to enable easy indexing of data and application model creation. We will use an Amazon EC2 instance to gain flexibility and ease in our deployment process. We are using technologies that our development team is familiar with to ensure a short and doable development timeline. Node JS along with the powerful Express JS framework will be used to keep a running server as well organize our routes to different pages. This will also be especially helpful when we make many HTTP requests to GET or POST data. We chose to use the Materialize CSS framework because it makes the styling fairly easy and responsive, and Angular JS to guide in that effort as well. We reserve the right to not use Materialize CSS however, should it make creating responsive elements harder. In that event, we will resort to simply using CSS media queries.

Chapter 9

Test Plan

9.1 Unit Testing

Unit testing of our website will be useful in the preliminary stages of testing. This form of testing can be done to ensure that we have correct functionality of individual features. The focus of this testing will be to ensure our web crawler grabs the data we specified it to. The database we decide to use will also be tested to ensure it can hold a high amount data entries as well as be reliable to grab entries efficiently. Lastly, our webpage will be thoroughly tested to ensure the integrity of the aesthetics is kept among several browsers and operating systems.

9.2 Integration Testing

The integration testing of our program will be one of the most crucial steps. During this testing phase we will begin connecting services and functionality to ensure that the program can function as a whole. Components will be pieced together one at a time so that the errors can be pinpointed and the problem can be specified. As each service is connected to the completed system the errors will become obvious as to which portion of the project is improperly functioning.

9.3 Alpha Testing

Alpha testing will be completed in-house by our team once integration testing is complete. During alpha testing we can test the completed system functionality as the product will be near completion and fully functional. As the system is tested, we will find the bugs and errors that are still prevalent in the code causing erroneous results. Along with catching the bugs in the program, we will also be able to ensure that all requirements are being satisfied by the system.

9.4 Beta Testing

During beta testing we will expand testing from only group members to friends, colleagues, or others interested in the project. By this step most user cases should be thoroughly tested and functional. We will use tester feedback to improve the user-interface aspects that are generally agreed upon. As some bugs will be still uncaught before this process, using beta testers will ensure that all possible situations are exhausted and that the program is ready to be used.

Chapter 10

Risk Analysis

10.1 Introduction

This section describes the several possible risks associated with the project. The first subsection contains descriptions of the different columns in the Risk Analysis table. The second subsection contains the table itself, which lists details such as the numerous risks and their associated levels of severity and consequences.

10.2 Table Description

Name of Risk: Specifically identifies specific risks associated with project.

Consequences: The outcome of the risks if they occur.

Probability: The likelihood of the risk occurring.

Severity: The effect the risk can impose on the project.

Impact: The probability multiplied by the severity. It determines a quantitative measurement of the risk on the project.

Mitigation Strategies: Possible actions to take to prevent the risks from occurring.

Risk	Consequences	Probability	Severity	Impact	Mitigation Strategies
Illness	Need to reassign remainder of work to the last remaining healthy group member to complete remainder of work. Pushed back deadlines and developmental process.	0.7	9	6.3	Get sufficient sleep throughout the process. Stay properly well nurse and hydrated. Take proper medications to leave it illnesses.
Inefficient time-planning	Project will be rushed leaving features incomplete or lacking functionality. Possibility of incomplete product by deadline. Project assignments may need to be that redistributed.	0.5	7	3.5	Stick to Gantt chart as to ensure deadlines are met along with proper progress of the project. Hold each other accountable to the deadline set. Proactively take on tasks tailored to one strengths.
Technical limitations	Unable to achieve desired functionality and User-interface. Project may lack key features and result in a slow or unusable interface.	0.4	8	3.2	Be upfront and honest about ability with the technology. As deadlines approaching ensure that proper steps are taken so that the assignment can be completed. Refresh through skills through on my tutorials and/or practic
Improper management of code	Complete loss of current version of code can occur if not properly stored or maintained. Current version of code can be manipulated to the point of losing key functionality with no version to restore from	0.2	7	1.6	Create a repository on Github to store code as well as keep a version of control in case roll backs are needed.
Unable to integrate functional components	Segmented code that holds functionality independently but fails to work together once combined. Key functions of code are not compatible with desired versions of other programs.	0.3	10	3	When deciding components of project make sure that they achieve desired functionality and also are compatible with one another.
Database/web storage system failures	Unable to store data into storage system. Data becomes unavailable for the site. Failure to retrieve data results in no video playback	0.2	9	1.8	Have a tool that checks that the storage system status is operational. Create checks to ensure data integrity.
Unable to acquire sufficient database of videos	Video selection will be severely lacking resulting in a small pool of videos for viewing. Difficulty in keeping current and attracting new users.	0.25	9	2.25	Explore known video and media outlets that constantly outp usable media. Carefully script information from the wedding is in a variety c keywords and phrases.
Video formats change/removed from host site	The removal or editing of videos on their host website creates dead links that will be unplayable on our website.	0.2	6	1.2	Create a program that tests old links within the database to ensure the video source is still live and usable.
Video copyright infringement	Video infringes on the copyrights resulting in the immediate removal of link	0.3	9	2.7	Crawl for videos from reputable websites that offer videos fo public use such as YouTube.
EC2 failure	Instance will fail and remain down.	0.3	4	1.2	Configure infrastucture to accomodate failures by preparing scripts to detect failures and provide additional instances.
Website URL offline	Unavailability of online access to website.	0.2	5	1	Ensure service and usage fees are paid up-to-date. Have an alert if website servers are down.

Figure 10.1: Risk Analysis Table

Chapter 11

Developmental Timeline

11.1 Introduction

This section covers our planned developmental timeline. We present our proposed plans via Gantt chart. The project was divided into major steps: project requirements, design, implementation, testing, and documentation. The tasks were dispersed on a per person basis.

	Fall Quarter (First Half)	Fall Quarter (Second Half)	Winter Quarter (First Half)	Winter Quarter (Second Half)	Spring Quarter (First Half)	Spring Quarter (Second Half)
Design						
Design UI			Deadline			
Design UI for different screen sizes			Deadline			
Implementation						
Front End Implementation				Deadline		
Front End Controller Implementation				Deadline		
Create DBs and Install NPM packages				Deadline		
Implement Backend (Recommendation System)				Deadline		
Implement Backend (Express Router)				Deadline		
Implement Backend (Web Crawler)				Deadline		
Testing						
Test UI / A/B Tests						Deadline
Test on Different Browsers						Deadline
Unit Testing						Deadline
Integration Testing						Deadline
Test on SCU Linux Machines						Deadline
Test on Different Screen Sizes						Deadline
Documentation						
Problem Statement	Deadline					
Requirement Compiling		Deadline				
Design Document		Deadline				
Final Presentation						Deadline
Final Report						Deadline

Figure 11.1: Gantt Chart