

---

# Project Final

for

## Cord Cutter Web Application

Version 4.0

Group Name: Team Two

Name	Role
Shaikh, Mohammed	Project Manager (PM)
Scobee, Michael	Test Director (TD)
Pede, Anthony	Software Designer (SD)
Owings, Edward	Requirements Manager (RM) Training Manager (UX) Alt - Software Designer (SD) Alt - Project Manager (PM)

Instructor: Terry Mentzos

Course: CMSC 495

Date: 11 Dec 2021

# Contents

<b>1</b>	<b>OVERVIEW .....</b>	
<b>2</b>	<b>PROJECT PLAN .....</b>	
2.1	SCOPE .....	1
2.1.1	<i>Project Justification .....</i>	
2.1.2	<i>Project Deliverable Outline .....</i>	
2.2	PROJECT CONTACTS .....	2
2.3	COMMUNICATIONS MANAGEMENT .....	2
2.4	RISK MANAGEMENT .....	3
2.4.1	<i>Risk Management Strategy .....</i>	
2.4.2	<i>Risk documentation .....</i>	
2.4.3	<i>Mitigation and Contingency Plans .....</i>	
2.5	CHANGE MANAGEMENT .....	4
<b>3</b>	<b>SOFTWARE REQUIREMENTS SPECIFICATION .....</b>	
3.1	INTRODUCTION .....	5
3.1.1	<i>Document Purpose .....</i>	
3.1.2	<i>Product Scope .....</i>	
3.1.3	<i>Intended Audience and Document Overview .....</i>	
3.1.4	<i>Definitions, Acronyms and Abbreviations .....</i>	
3.1.5	<i>Document Conventions .....</i>	
3.2	OVERALL DESCRIPTION .....	6
3.2.1	<i>Product Overview .....</i>	
3.2.2	<i>Product Functionality .....</i>	
3.2.3	<i>Design and Implementation Constraints .....</i>	
3.2.4	<i>Assumptions and Dependencies .....</i>	
3.3	SPECIFIC REQUIREMENTS .....	8
3.3.1	<i>External Interface Requirements .....</i>	
3.3.2	<i>Functional Requirements .....</i>	
3.3.3	<i>Use Case Diagram: .....</i>	
3.3.4	<i>Security Requirements .....</i>	
3.3.5	<i>Other Requirements .....</i>	
3.3.6	<i>Additional requirements outside functional scope: .....</i>	
<b>4</b>	<b>USER GUIDE .....</b>	
4.1	PREREQUISITES TO USING A STREAMING SERVICE .....	10
4.1.1	<i>Internet .....</i>	
4.1.2	<i>Streaming Device .....</i>	
4.1.3	<i>Assumptions About Your Home .....</i>	
4.1.4	<i>Cost Estimate Table .....</i>	
4.2	USING THE CORD CUTTER WEB APP .....	11
4.2.1	<i>Navigate to the website .....</i>	
4.2.2	<i>Enter your details .....</i>	
4.2.3	<i>Select your TV Provider .....</i>	
4.2.4	<i>Select your Channel Package .....</i>	
4.2.5	<i>Confirm and Submit .....</i>	
4.2.6	<i>Analyze and Print/Save your Report .....</i>	
4.2.7	<i>Submit A Help Ticket or Question .....</i>	
<b>5</b>	<b>TEST PLAN .....</b>	
5.1	INTRODUCTION .....	14
5.1.1	<i>Purpose of the test plan document .....</i>	
5.2	COMPATIBILITY TESTING .....	14

5.2.1	test risks / issues .....	
5.2.2	items to be tested / not tested .....	
5.2.3	Test approach(s) .....	
5.2.4	Test pass / fail criteria .....	
5.2.5	Test entry / exit criteria .....	
5.2.6	Test deliverables .....	
5.3	USER INTERFACE TESTING.....	15
5.3.1	Test risks / issues .....	
5.3.2	Items to be tested / not tested .....	
5.3.3	Test approach(s) .....	
5.3.4	Test pass / fail criteria .....	
5.3.5	Test entry / exit criteria .....	
5.3.6	Test deliverables .....	
5.4	FUNCTIONAL TESTING.....	16
5.4.1	Test risks / issues .....	
5.4.2	Items to be tested / not tested .....	
5.4.3	Test approach(s) .....	
5.4.4	Test pass / fail criteria .....	
5.4.5	Test entry / exit criteria .....	
5.4.6	Test deliverables .....	
5.5	UNIT TESTING .....	17
5.5.1	Test risks / issues .....	
5.5.2	Items to be tested / not tested .....	
5.5.3	Test approach(s) .....	
5.5.4	Test pass / fail criteria .....	
5.5.5	Test entry / exit criteria .....	
5.5.6	Test deliverables .....	
5.6	END-TO-END TESTING .....	18
5.6.1	Test risks / issues .....	
5.6.2	Items to be tested / not tested .....	
5.6.3	Test approach(s) .....	
5.6.4	Test pass / fail criteria .....	
5.6.5	Test entry / exit criteria .....	
5.6.6	Test deliverables .....	
5.7	TEST TEAM.....	19
5.7.1	Members and roles.....	
5.8	PRODUCTIVITY TOOLS.....	20
5.8.1	JIRA .....	
5.8.2	Testing .....	
5.9	DATA REQUIREMENTS.....	20
<b>6</b>	<b>DESIGN &amp; ALTERNATE DESIGNS .....</b>	
6.1	INTRODUCTION.....	20
6.1.1	Purpose of the Product Design Specification Document .....	
6.2	GENERAL OVERVIEW AND DESIGN GUIDELINES/APPROACH .....	21
6.2.1	Assumptions .....	
6.3	ARCHITECTURE DESIGN .....	21
6.3.1	Hardware Architecture .....	
6.3.2	Software Architecture .....	
6.3.3	Communication Architecture .....	
6.3.4	Performance .....	
6.4	SYSTEM DESIGN .....	23
6.4.1	Use-Cases .....	
6.4.1	Backend/Classes Design .....	
6.4.2	Database Design .....	

**7 VERSION HISTORY**

7.1	INTRODUCTION .....	25
7.2	VERSION HISTORY .....	26
7.2.1	Version 0.0 .....	
7.2.2	Version 1.0 (No Comments) .....	
7.2.3	Version 2.0 (No Comments) .....	
7.2.4	Version 3.0 (No Comments) .....	
7.2.5	Version 4.0 (Final) .....	

**8 CONCLUSIONS**

8.1	INTRODUCTION .....	27
-----	--------------------	----

**APPENDIX**

APPENDIX: REFERENCES .....	27
APPENDIX: KEY TERMS .....	28
APPENDIX: FILE ATTACHMENTS .....	29
<i>Timeline</i> .....	
<i>User Guide</i> .....	
APPENDIX: TEST REPORT .....	30

## Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Edward Owings	Drafted SRS plan using IEEE template.	10/30/2021
1.1	Edward Owings	Integrated project plan and wrote overview.	10/31/2021
1.3	Edward Owings	completed required project plan sections.	10/31/2021
1.5	Mohammed Shaikh	Added details in overview, requirements, functionality, design and implementation constraints, functional requirements.	11/1/2021
2.0	Edward Owings	Added all required sections and incorporated test plan from week three into the document.	11/28/2021
3.0	Edward Owings	Updated sections to show changes and modifications in code.	12/05/2021

# 1 Overview

The project described within this document is a web application designed with the specific goal of helping cable television subscribers cancel their cable subscription and sign up for streaming services. The application is designed so that the user can make an informed decision on cost, and quality of programming before they depart their cable provider. The user will be presented with a user interface to compare existing streaming services, but no actual data is committed, meaning the user can only retrieve data and not put/delete any data.

The application is targeted at the cable television market because current cable prices have been slowly getting higher as major television networks focus more on their streaming platforms. This application will allow those that are stuck paying higher costs for the programs they watch to have a more affordable option.

This document gives a preliminary plan for how team two aims to implement the product. The project plan wholly will be a living document throughout development and will become final when it is put into production for grading by Professor Mentzos. The goal of this plan is not only to guide the team in a mapped-out manner toward a final product but to also demonstrate the ability to execute a project plan within the Software Development Life Cycle, practice finding risks, defining acceptance criteria, high-level test planning, and define technical specifications.

# 2 Project Plan

## 2.1 Scope

### 2.1.1 Project Justification

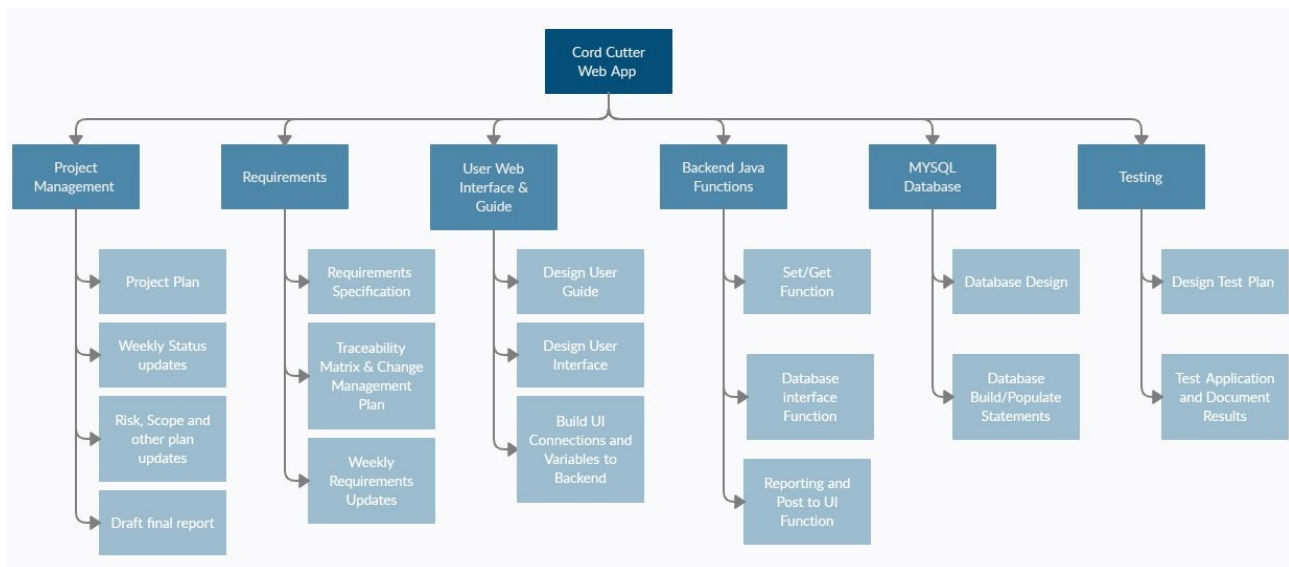
The true purpose of the Cord Cutter Web Application is to demonstrate this team's ability to accomplish a software development project from cradle to production. This project will be complete at the production phase, but it is known that a typical SDLC project plan will continue until it is retired. This plan is intended to satisfy the requirement of having a project plan, all sections within are from the provided 'required' template materials given in week two by the professor.

### 2.1.2 Project Deliverable Outline

Each task in this project is broken down into categories that correspond with a team members role. These tasks are then broken down further onto the milestone table to view how and when the tasks will be due to the customer. When each deliverable is completed, the same deliverable will be submitted amongst all team members.

#### 2.1.2.1 Deliverables

All work has been divided up based on team member roles, as shown in the below figure. The milestones break apart each of the tasks into weekly assigned tasks making the overall project spread out evenly. These milestones can be viewed in the Appendix [Team Timeline](#)



## 2.2 Project Contacts

This table is a directory of individuals who play a significant role or can have significant impact on this project.

**Table 1 - Contacts**

Role	Name	E-mail Address
Project Sponsor	Terry Mentzos	Terry.mentzos@faculty.umgc.edu
Project Manager	Shaikh, Mohammed	Mshaikh2@student.umgc.edu
Test Director	Scobee, Michael	Mscobee90@gmail.com
Software Designer	Pede, Anthony	Anthonypede@gmail.com
Requirements Manager/ Training Manager	Owings, Edward	eowings@student.umgc.edu

## 2.3 Communications Management

This table lists the different communication items needed for this project. A communication item may be a Word document, an e-mail, a meeting, or anything called for by another plan.

**Table 2 - Communications Plan**

What	When	How	Responsible	Approval	Audience
Project Kickoff	Project Start (include date when planned)	Meeting	Project Manager	Approve	Request comments

Team Meetings	Weekly	Slack	ALL		
Major Milestone Announcements	As completed	Jira/Slack	ALL		

## 2.4 Risk Management

### 2.4.1 Risk Management Strategy

Risks will be identified in weekly team discussions on a specific task and their severity will be determined using the below chart. Once severity is identified the risk will be documented on the risk table. Once risks are documented a mitigation plan will be agreed upon by the entire team these plans will be documented in the mitigation table. Once risks have an agreed mitigation the risk will be reevaluated in the mitigated risk table. All team members will be responsible of ensuring any risks that need to be mitigated are done so within the confines of the agreed upon plan documented in this risk management strategy section.

**Table 3 - Risk Severity Matrix**

Probability	Harm severity			
	Negligible	Marginal	Critical	Catastrophic
Certain	High	High	Very high	Very high
Likely	Medium	High	High	Very high
Possible	Low	Medium	High	Very high
Unlikely	Low	Medium	Medium	High
Rare	Low	Low	Medium	Medium
Eliminated	Eliminated			

### 2.4.2 Risk documentation

When a risk is discovered, it shall be documented here so that it can be examined for mitigation.

**Table 4 - Risk Table**

#	WBS Task	Risk Description	Severity
1	Project Plan	Plan does not get submitted	High

### 2.4.3 Mitigation and Contingency Plans

Enter the mitigation activities that can be accomplished to prevent the risk from happening.

**Table 5 - Mitigation Table**

Risk #	Mitigate
1	<ul style="list-style-type: none"> <li>Communicate with role responsible and confirm they will not get in on time.</li> <li>Break deliverable up between participating team members</li> <li>Submit deliverable</li> </ul>

**Table 6 - Mitigated Risk Table**

#	WBS Task	Risk Description	Severity
1	Project Plan	Plan does not get submitted	Medium

## 2.5 Change Management

Changes will be categorized as standard urgent and normal and will be addressed as they arise during development. Team two will use standard SDLC change management processes. If an urgent change arises a single team, member may enact this change without approval of the team. All other changes should be discussed and agreed upon for approval.



Table 7 - Change Management

Change	Reason + Approver Initials	Date of Change	Category
<b>Example:</b> Changed code in reports function.	Full work stoppage due to function not executing. Added import of jasper reports library, reports function now working as intended. ELO	10/30/2021	<b>Urgent</b>

## 3 Software Requirements Specification

### 3.1 Introduction

This software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for the Cord Cutter Web Application which is currently under development. Furthermore, this SRS will describe what the software will do and how it will be expected to perform as well as the functions required.

#### 3.1.1 Document Purpose

This Software Requirements Specification (SRS) will include the project needs to make the application fully functioning and usable. Items listed herein will give details on product and deliverables. This is a living document and will be updated throughout development and into testing and production.

#### 3.1.2 Product Scope

The cord cutter web application is designed to allow the consumer the ability to make an educated decision on alternatives to cable television. This application will be hosted on a server and accessible to anyone who has access to the world wide web.

This application will benefit the consumer because the cable television market is shrinking thus driving up the costs to cable television. This application will exceed the consumers needs in comparing their current services with that of available streaming service.

#### 3.1.3 Intended Audience and Document Overview

Project manager: The project manager will ensure the needs listed within this document line up with the needs of the project as a whole. This document should compliment and integrate wholly into the project plan.

Developers: The developers of the web application will utilize this document as a map and timeline to the project in reference to what the project needs. This document should compliment and integrate wholly into the development plan.

Testers: Testers will use this document in the same manner as developers yet in regard to developing a testing plan for the project. This document should compliment and integrate wholly into the test plan.

Client (Professor): This document will show the client what and when the project will be reaching certain milestones and the specific requirements the project will need at each phase.

This document should be initially read though each section fully from start to finish. Any subsequent reads can be tailored to information being sought after. This SRS will cover the project requirements based on functional and non-functional categories.

### **3.1.4 Definitions, Acronyms and Abbreviations**

AWS: Amazon Web Services

SQL: Structured Query Language

### **3.1.5 Document Conventions**

This document follows the IEEE formatting requirements. Use Arial font size 11, or 12 throughout the document for text. Use italics for comments. Document text should be single spaced and maintain the 1" margins found in this template. For Section and Subsection titles please follow the template.

## **3.2 Overall Description**

### **3.2.1 Product Overview**

This project will be self sustaining and operatable within its own environment. This means it is basically not a smaller part of a larger application or software suite. The web application will start at the landing page where the user will select their zip code using a geospatial chart. From which will direct the user to select their cable tv provider. From there the user will need to choose the channel package they subscribe to. The rest of the interactions are done from the application on the back end. The back end of the application will take in the user inputs and process them off a mysql database to compare streaming services with those the user subscribes to on their cable plan. The output will break prices down and allow the user to filter into segmented categories. Once the user has made any final edits to what they which services to retain and which to discard the application will then produce a report detailing the streaming services, how to purchase and total costs. None of the actual subscriptions are made within the application itself, and everything stored in the database will be set attributes of each cable provider. However, input validation shall be implemented so bad data is not inserted which can cause the application to break. Included with input validation should be protection against sql injection, so that a malicious user can't insert bad data and lead other users to get their information stolen.

### **3.2.2 Product Functionality**

**User Interface:** to present all data to the client in a neat, structured manner will use combination of HTML, CSS.

**Database:** This will be where all the channels are stored. The structure needs to allow both cable tv services and streaming services to interact with each channel independently the database will be MYSQL. Data types will be outlined and enforced within the UI and database constraint definitions, so there are no errors upon any input that can break the application.

**Backend:** This will be the computation and interface between the user interface and the database. Will use JAVA Spring Boot as a framework.

### **3.2.3 Design and Implementation Constraints**

All team members should have the same IDE installed with all mirroring libraries, so all development has the same dependencies. IDEs such as Xcode should be avoided since it is solely used on Mac machines. Even if all team members have Mac, we should be in a position so that if the machine is not readily available, we can pick up where we left off if we need to borrow a machine or use a shared machine.

AWS funding uncertainty, the app will be served from an AWS virtual machine for final testing and production only. The costs must consume less than the available AWS funds which at start is \$47.00. The fall-back plan in case funds are consumed before final production will be serving the app on a Raspberry pi hosted from student home.

Delivering the project code via JAR file to professor for periodic deliverables may cause errors in executing the app based on settings stated for server configuration. Spring boot should cover this limitation and run seamlessly but this still has not been tested hence it being listed here as a constraint.

This project plan and design will be implemented using the software development lifecycle process of Agile. Sprints are divided into weekly sprints since the project has weekly deliverables.

With the combination of scope, time, and cost,

### **3.2.4 Assumptions and Dependencies**

Each team member will be available to complete any deliverables that are inherited by their role. Furthermore, all team members will assist in other team members who have communicated in a timely manner they are struggling with their role.

Budgetary limitations of AWS are assumed to afford the development and delivery of the completed project.

The users web browser is assumed to be compatible with the web application.

Any dependencies will be listed as they arise but currently the all-encompassing spring boot framework should make the dependencies very low.

There may be a need to have [Apache2 server running on the front end of spring boot](#)

## 3.3 Specific Requirements

### 3.3.1 External Interface Requirements

#### **3.3.1.1 User Interfaces**

The user will interface with the web application via their browser (chrome, edge, Firefox, opera...), the user experience plan will cover detailed user interactions and commonly asked questions.

#### **3.3.1.2 Hardware Interfaces**

The spring boot application will need to be hosted on an AWS virtual EC2 machine which is depended on the AWS physical server farms. There are no interactions between hardware functionality and software in this application. For example, this app will not switch something on or maneuver something robotically. The main hardware dependency is that of the server the app is hosted on and the machine the user is accessing the app from.

#### **3.3.1.3 Software Interfaces**

The application will use HTTP request methods securely such as GET or POST to take info from the user. The back end of the application will then compute those data blocks and use data within multiple tables to configure a response. The response will then be delivered to the user with the same secure HTTP request methods. All software languages and types within the communication chain include JAVA, HTML, MYSQL, CSS.

### 3.3.2 Functional Requirements

NOTE – these functions are expected to change as the project matures and currently are a 'bare bones' estimation of the functions needed in the web application.

Below are the functions that make up the system as a whole. Each function should have its own individual action or abilities. This list is expected to change fluidly through to production.

#### **3.3.2.1F1: Web interface requirements:**

- Must have all selection options needed such as zip code map, cable provider.
- Must be aesthetically pleasing to look at using CSS style sheet.
- Must include secure code that does not allow backend malicious interactions.
- Must use input validation to ensure malicious input is prevented.
- Must use HTTPS to ensure secure network connection.

#### **3.3.2.2F2: Java Getter/Setter function:**

- Must assign user inputs to arrays, numerical, string, computational variables to be passed within the application.
- Ensure consistent naming conventions to easily identify a setter function and getter function

#### **3.3.2.3F3: Java main function:**

- Must compute and process the users inputs against the database and output to multiple functions and interface with the database.

**3.3.2.4F4: Java Report function:**

- Must use the resulting data to output a report that is organised and easy to read by user
- If possible, make report customizable so user can arrange it in multiple output formats.
- Ensure there is no page cut-off upon report generation

**3.3.2.5F5: Java database interface function:**

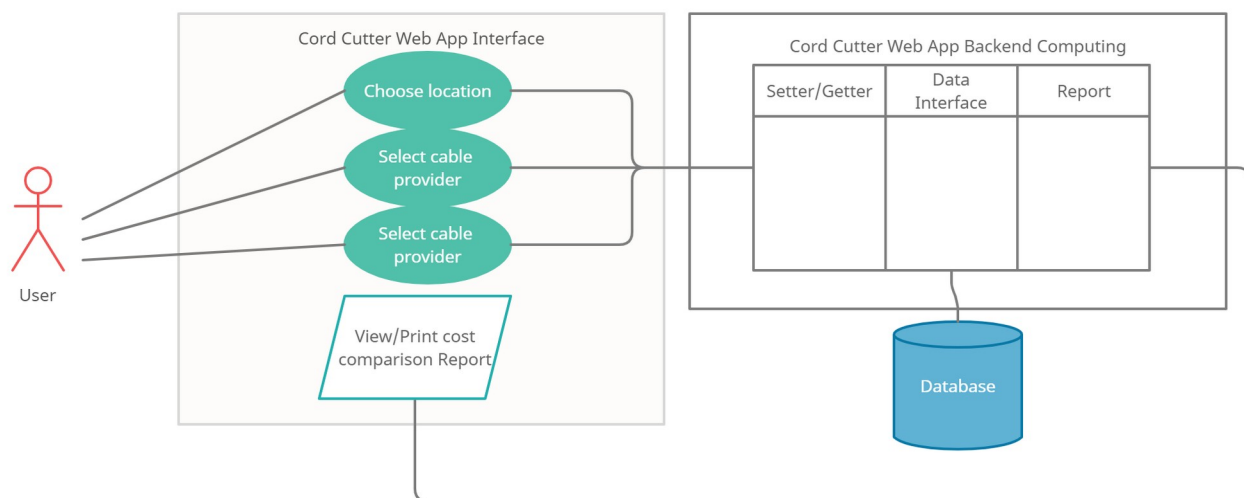
- Must interact both input and output with database and build data into a reportable format.
- Designed with security requirements to ensure bad data is not permissible

**3.3.2.6F6: MYSQL Database**

- Must draft all table design to integrate flawlessly with the application.
- Table creating statements should be drafted and saved to text files.
- Table populating statements containing cable provider & streaming service data should be drafted and saved to text files.
- Define data types for each field
- Ensure secure login credentials, including complex password and secure port

**3.3.2.7F7: Test function(multiple):**

- Each test function must test the function it is drafted under.
- Test functions must be kept in mind as drafting primary functions to ensure full testability of each function's capabilities.
- TestNG framework should be used, since it is all-inclusive in test definitions and test reporting.

**3.3.3 Use Case Diagram:****Figure 2 – Use Case Diagram**

### 3.3.4 Security Requirements

*Security requirements currently foreseeable would be protecting the application from database injection, cross site scripting and any other user input attacks.*

### 3.3.5 Other Requirements

### 3.3.6 Additional requirements outside functional scope:

Team will need to hold a meeting prior to week five to configure and set up the eclipse IDE to be operational on all team members computers.

## 4 User Guide

### 4.1 Prerequisites to Using a Streaming Service

Users must be aware that before switching from cable television to a streaming service there are multiple prerequisite purchases that must be made. This application will not include these costs, but if you populate the table below with the items of your choosing and cost you will have a good estimate of these costs.

#### 4.1.1 Internet

If you have an internet service great if you don't then you will need one. The recommended internet speed for high-definition television is 25 Mbps. If you are unsure of your internet's connection speed please go to the following URL and check your speed, <https://www.speedtest.net/> If you currently have faster than 25Mbps great, if you do not have any internet or it is lower than 25Mbps you will need to acquire/upgrade internet services. The current average internet plan for 25Mbps is \$35.00 per month.

#### 4.1.2 Streaming Device

There are many options for streaming devices, ranging in costs from a few dollars to thousands of dollars. If you have a smart TV, you may not have the full options offered by this web application as some smart televisions do not offer all streaming services. The most compatible devices you can get for streaming television is one of the four following devices: FireTV, AppleTV, Chromecast or Roku. Each of these devices can be googled and each offers a wide variety of technology the minimal option of each will produce a comparable image to your current cable box. Estimated costs for the most common 'stick style' streaming device is currently valued at \$50.00.

#### 4.1.3 Assumptions About Your Home

Besides the previous prerequisites this guide will assume the following about your household television set-up.

It is assumed that:

- Your television has an HDMI port
- You have cable TV (This app only compares streaming vs cable TV)
- You have an available 110v outlet or USB to power your streaming device.

#### 4.1.4 Cost Estimate Table

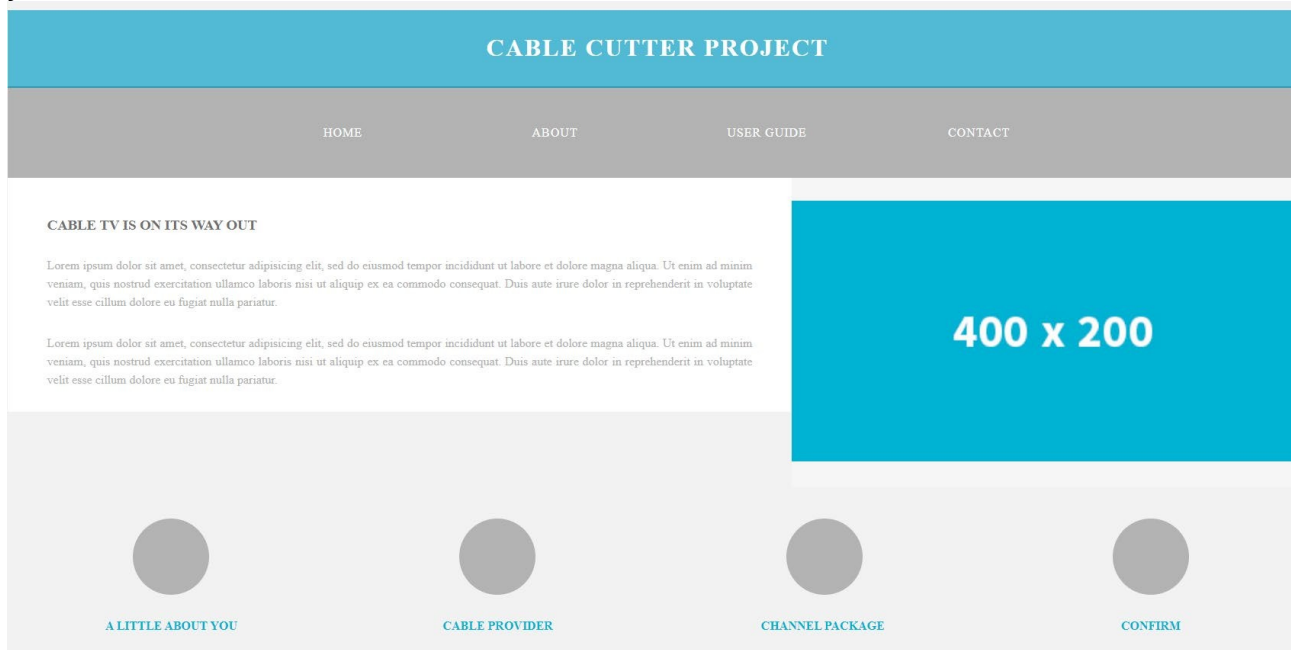
Use the table below to help estimate your costs. These costs can be added to your comparison report produced from the app to produce a more realistic cost comparison. Below you can enter your new cost followed by old cost then calculate the difference and record the number. After each row has been accomplished you can then take the sum for all your costs for the result or total.

	New cost	Old Cost	New Minus Old Cost
Internet			
Streaming device			
Other			
		Total	

## 4.2 Using the Cord Cutter Web App

### 4.2.1 Navigate to the website

First you need to navigate to the cord cutter web application (<https://cmsc495t2.pages.dev/>) using your default web browser as seen below.



### 4.2.2 Enter your details

First you will tell us your first name and email:

**Your Info**

First name:

E-mail:

Cable Provider:

FIOS

Channel Package:

The Most FiOS TV

Submit

Reset

#### 4.2.3 Select your TV Provider

Now you must choose your cable provider currently we only compare FiOS, Cox and Comcast.

Cable Provider:

FIOS

FIOS

Cox

Comcast

#### 4.2.4 Select your Channel Package

Now you need to select the package you subscribe to. Please ensure it aligns which a offering from your cable provider.

Channel Package:

The Most FiOS TV

The Most FiOS TV

More FiOS TV

Contour TV Starter

Contour TV Preferred

Contour TV Preferred Plus

Contour TV Ultimate

STANDARD+

SELECT+

SIGNATURE+

SUPER+



#### 4.2.5 Confirm and Submit

Here you can ensure all your entries are correct and if so click 'Submit' if you wish you may also 'Reset' all fields.

**If choices are correct please press continue, if choices are not correct press back to return to form.**

Full name: TestUserDec04  
E-mail: Email@email.com  
Cable Provider: fios  
TV Package: FIOS2

Continue

Back

#### 4.2.6 Analyze and Print/Save your Report

This final step you will be redirected to your cost/channel comparison report. This report will show you any savings you will see if switching to streaming services and if you will have all your current programming options.

TV ON TV	Cost	Cost	Cost	Cost	Cost
Monthly Cost Comparison	\$85.00	\$64.99	\$25.00	\$50.00	\$64.99
Channel Count Comparison	64	64	39	65	61

#### SAVINGS

Streaming Provider	Monthly Savings
Fubo	\$21
Philo	\$60
Sling	\$35
Youtube	\$21

Print Page

\* Note most streaming services offer live tv options but this comparison will not show weather or not this is offered.

#### 4.2.7 Submit A Help Ticket or Question

From the web applications navigation menu select the 'Contact' option. This will take you to the contact form as show below:

First Name

Enter your first name

E-Mail Address

Enter your email address

Message

Enter your message.

Submit

Sun, December 05 2021

14:52:03, UTC

## 5 Test Plan

### 5.1 Introduction

#### 5.1.1 Purpose of the test plan document

This document serves to identify what each team member will be responsible for during the testing phase, what tools will be used, detail data requirements, as well as specifying what will and will not be tested, and how those tests will be executed.

### 5.2 Compatibility testing

#### 5.2.1 test risks / issues

This application is web-based and there is the potential for many different types of browsers to be used to access it. This can run the risk of certain browsers not being compatible with the application and testing will need to be conducted to address this concern.

#### 5.2.2 items to be tested / not tested

**Table 8 – Outside Software Test Items**

Item to Test	Test Description	Test Date	Responsibility
Google Chrome	Test the compatibility of the application on Chrome	12/06/2021	Scobee, Michael
Mozilla Firefox	Test the compatibility of the application on Firefox	12/06/2021	Scobee, Michael

Microsoft Edge	Test the compatibility of the application on Edge	12/06/2021	Scobee, Michael
Opera	Test the compatibility of the application on Opera	12/06/2021	Scobee, Michael

### 5.2.3 Test approach(s)

In order to test the compatibility of each browser, testers will execute end-to-end testing within each browser to ensure that no errors are found.

### 5.2.4 Test pass / fail criteria

A test pass will occur when an end to test is successfully executed on the browser. Should any bug/error occur during this end-to-end test then the test will be considered a failure.

### 5.2.5 Test entry / exit criteria

The entry criteria is a functioning application that allows for end-to-end testing, meaning a functioning UI, database, and queries written to pull and compare data. The exit criteria are when all four browsers can finish testing with no errors.

### 5.2.6 Test deliverables

Test deliverables include test cases, test data, test summary reports, and test closure reports that conform to IEEE standards.

## 5.3 User interface testing

### 5.3.1 Test risks / issues

The user interface is the key component that allows users to use this application and if any of the functioning aspects are not working as intended it will be an issue. One such risk is interface buttons not working correctly for the user. It is also expected to be structured in a manner that easily represents data to the user.

### 5.3.2 Items to be tested / not tested

**Table 9 - UI Test Items**

Item to Test	Test Description	Test Date	Responsibility
UI Inputs	Test zip code inputs among others if	11/11/2021	Owings, Edward

	applicable		
Widgets	All widgets are functioning as expected.	11/11/2021	Owings, Edward

### 5.3.3 Test approach(s)

To properly test the user interface testers will undergo end-to-end testing of the software to ensure that no issues arise within the user interface that

### 5.3.4 Test pass / fail criteria

Tests will pass when an end-to-end test can be conducted without issues in the user interface, and fail otherwise.

### 5.3.5 Test entry / exit criteria

The entry criteria is a functioning user interface application and exit criteria are no P3 issues found within the UI testing.

### 5.3.6 Test deliverables

Test deliverables include test cases, test data, test summary reports, and test closure reports that conform to IEEE standards.

## 5.4 functional testing

### 5.4.1 Test risks / issues

Functional testing is intended to be automated with TestNG, this is a framework not all testers are familiar with and could require additional time for testers to become accustomed to using it.

### 5.4.2 Items to be tested / not tested

**Table 10 - Functional Test Items**

Item to Test	Test Description	Test Date	Responsibility
Zip Code API / IP	Test that zip code data that is automated is accurate.	11/16/2021	Pede, Anthony
Price Breakdowns	Validate price breakdowns of both streaming services and cable providers	11/16/2021	Pede, Anthony

	are accurate.		
Available services	Correctly displays all available cable providers based on users' zip codes.	11/16/2021	Pede, Anthony
Session	Since accounts are not being used test that sessions are working correctly and timing out after a given time.	11/18/2021	Pede, Anthony
Report	Final report to the user accurately representing the comparison.	11/22/2021	Pede, Anthony

### 5.4.3 Test approach(s)

Functional testing will be used to test the main functions of this application to check for usability and errors.

### 5.4.4 Test pass / fail criteria

Test passes will occur if the main functions are working without errors or usability concerns.

### 5.4.5 Test entry / exit criteria

Functional testing will begin when the development team has the main functions of the application in a usable state.

### 5.4.6 Test deliverables

Test deliverables include test cases, test data, test summary reports, and test closure reports that conform to IEEE standards.

## 5.5 Unit testing

### 5.5.1 Test risks / issues

The unit tests will be conducted using TestNG, a framework that not all testers are familiar with and may require additional time to learn how to write and execute tests.

### 5.5.2 Items to be tested / not tested

**Table 11 - User Input Test Items**

Item to Test	Test Description	Test Date	Responsibility
User Inputs	Test input from user for name, email, zip code.	12/1/2021	Shaikh, Mohammed
Provider Selection	Test selection options for cable providers	12/1/2021	Shaikh, Mohammed
Package Selection	Test package selection for cable provider	12/1/2021	Shaikh, Mohammed
Submit/Report	Test submission and report provided	12/1/2021	Shaikh, Mohammed

### 5.5.3 Test approach(s)

Use unit testing with TestNG, which will run input tests to ensure proper data types are accepted, and invalid data triggers errors. The unit tests will assert that an error message exists whenever a negative test is executed. An HTML report will automatically be generated upon completion of tests.

### 5.5.4 Test pass / fail criteria

All unit tests should pass with no issues. Any issue that is discovered as a result of a unit test will be considered a P3 defect, to be fixed before launch.

### 5.5.5 Test entry / exit criteria

All requirements should be implemented before testing commences. There should be no defect that doesn't pass an acceptance criteria/requirement.

### 5.5.6 Test deliverables

Test deliverables include test cases, test data, test summary reports, and test closure reports that conform to IEEE standards.

## 5.6 End-to-end testing

### 5.6.1 Test risks / issues

Since this is a small application hosted on the web, end-to-end testing is minimal, as this application will not be interacting with other applications. Feedback will be immediate so there is no anticipated waiting period to execute all tests. For the Cord Cutter application, the only risks that exist is if the web server is not up and running, and if there are connectivity issues with the database.

### 5.6.2 Items to be tested / not tested

**Table 12 - Final Test Items**

Item to Test	Test Description	Test Date	Responsibility
User experience	Test the application in its entirety to simulate a user trying the application.	12/5/2021	Scobee, Michael

### 5.6.3 Test approach(s)

End-to-end testing for overall user experience.

### 5.6.4 Test pass / fail criteria

Successful use of the application with no errors.

### 5.6.5 Test entry / exit criteria

Test entry will begin once all other forms of testing have been completed.

### 5.6.6 Test deliverables

Test deliverables include test cases, test data, test summary reports, and test closure reports that conform to IEEE standards.

## 5.7 Test team

### 5.7.1 Members and roles

Member	Role
Shaikh, Mohammed	Unit Testing
Scobee, Michael	UI Testing, Compatibility Testing

Pede, Anthony	End to End Testing
Owings, Edward	Functional Testing

## 5.8 Productivity tools

### 5.8.1 JIRA

Jira is an issue and tracking software that will be used to report, detail, and resolve bugs found within the application. When a bug is found it can be reported by creating an issue on Jira from the project home page. Each issue will be given a type, in this case, "Bug", as well as a brief description and a priority. Documentation for the setup and use of Jira can be found at <https://confluence.atlassian.com/jira/jira-documentation-1556.html>.

The priorities of bugs include P1 - Hotfix, P3 - To be fixed prior to launch, P5 - low priority, is not required to be fixed. P1 is not used as the application is not yet in production.

### 5.8.2 Testing

TestNG is a testing framework for Java similar to that of JUnit. It is used to perform tests for end-to-end, unit, function, and several others. There already exists support for TestNG in the Eclipse IDE that the project team is using, allowing for easy setup. One of the advantages of TestNG is that it generates reports on each test case for easy viewing of how the test went. Documentation can be found at <https://testng.org/doc/>.

## 5.9 Data requirements

### 9.1 Data

Data requirements for this project include available cable providers, what package each cable provider has available with their available channels and costs, as well as streaming service costs and content provided.

# 6 Design & Alternate Designs

## 6.1 Introduction

### 6.1.1 Purpose of the Product Design Specification Document

This section formally is intended to document design specifications and ideas. Note the intent of this



section is not the same as the detailed design in section four. This section will be used as a reference that the project manager and design team can use and update to document the design process and ideas.

## 6.2 General Overview and Design Guidelines/Approach

This section describes the principles and strategies to be used as guidelines when designing and implementing the system.

### 6.2.1 Assumptions

It is assumed that the design team has a fully configured operating IDE in which code will be ran. tested and written.

## 6.3 Architecture Design

The cord cutter web application will be hosted on a Tomcat 9 webserver using a ubuntu operating system. The web app will be written in Java and will require java 11 and spring boot plus many other dependencies that will be listed in the detailed design and .pom file.

The user will interact with the application using a Java server page html interface. This interface will pass user inputs to the MVC controller to be handled according to the specified action on the interface page.

### 6.3.1 Hardware Architecture

The simple hardware design requires the user's hardware to run a browser, the raspberry pi hardware to run the server and the AWS database hardware to run the database.

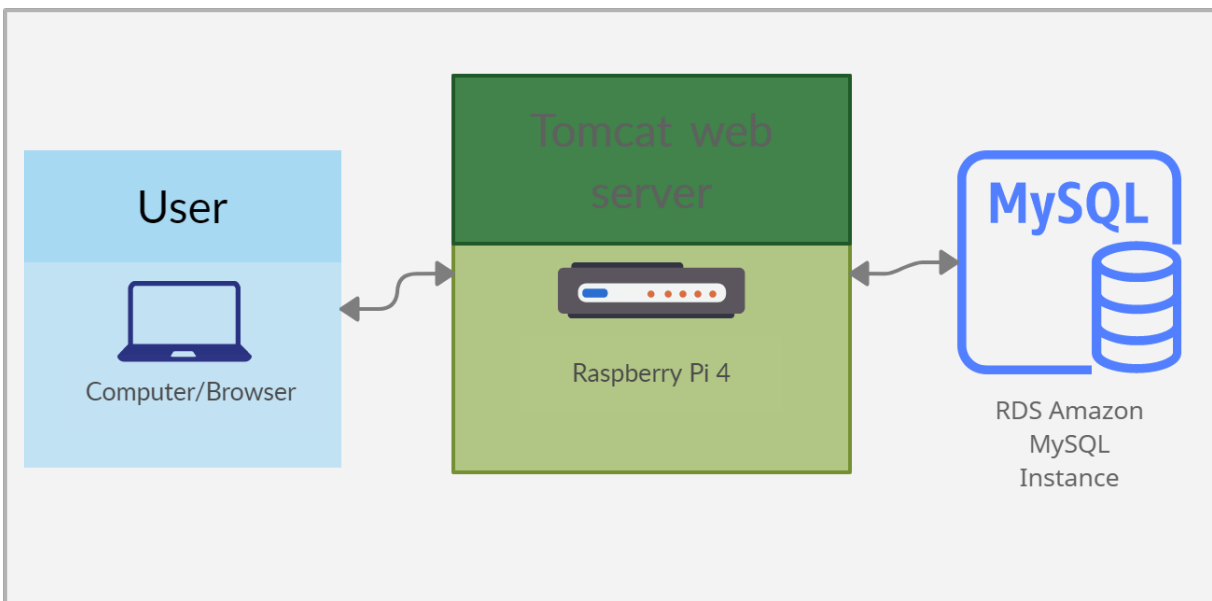


Figure 3 - Hardware Diagram

### 6.3.2 Software Architecture

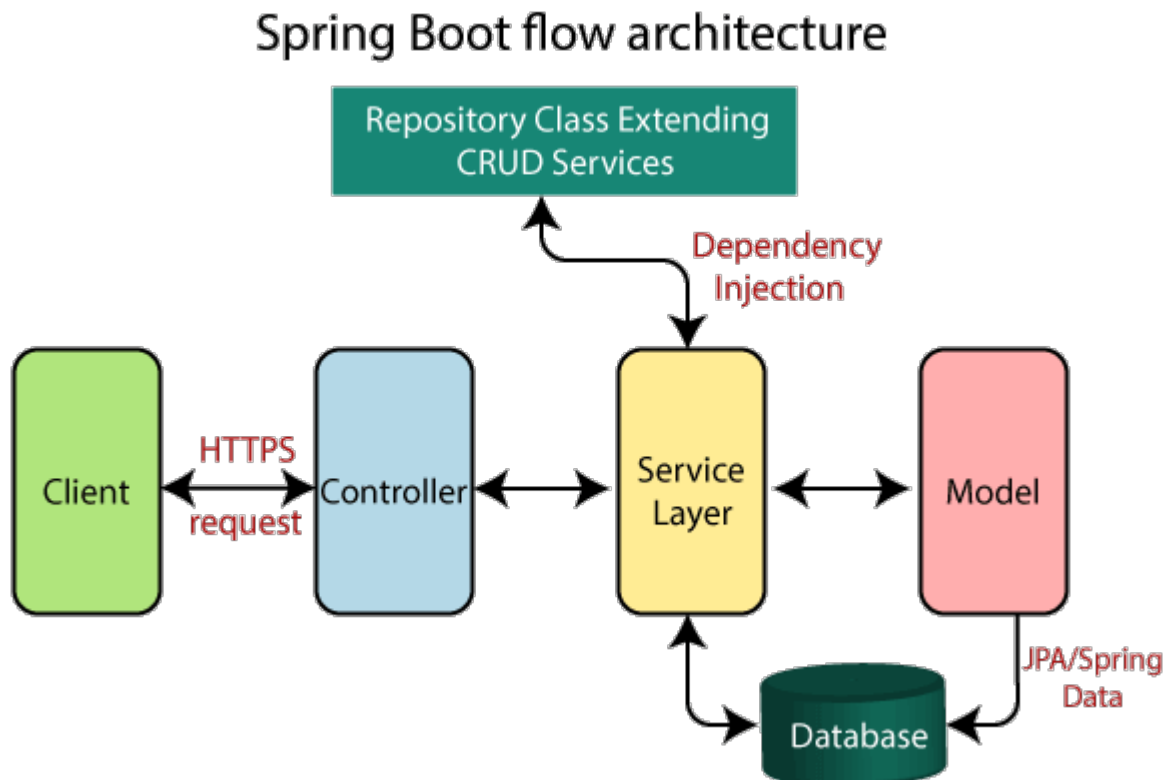


Figure 4 - Software Diagram

### 6.3.3 Communication Architecture

All communications within the application occur from the Model MVC

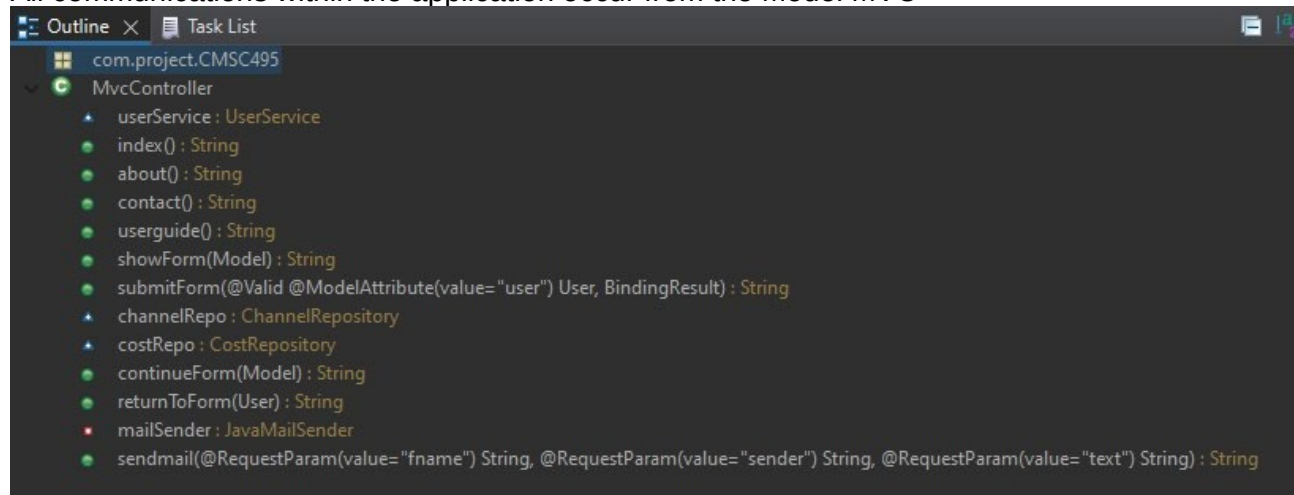


Figure 5 - Controller Diagram

### 6.3.4 Performance

Database performance dashboard readout after one month uptime.

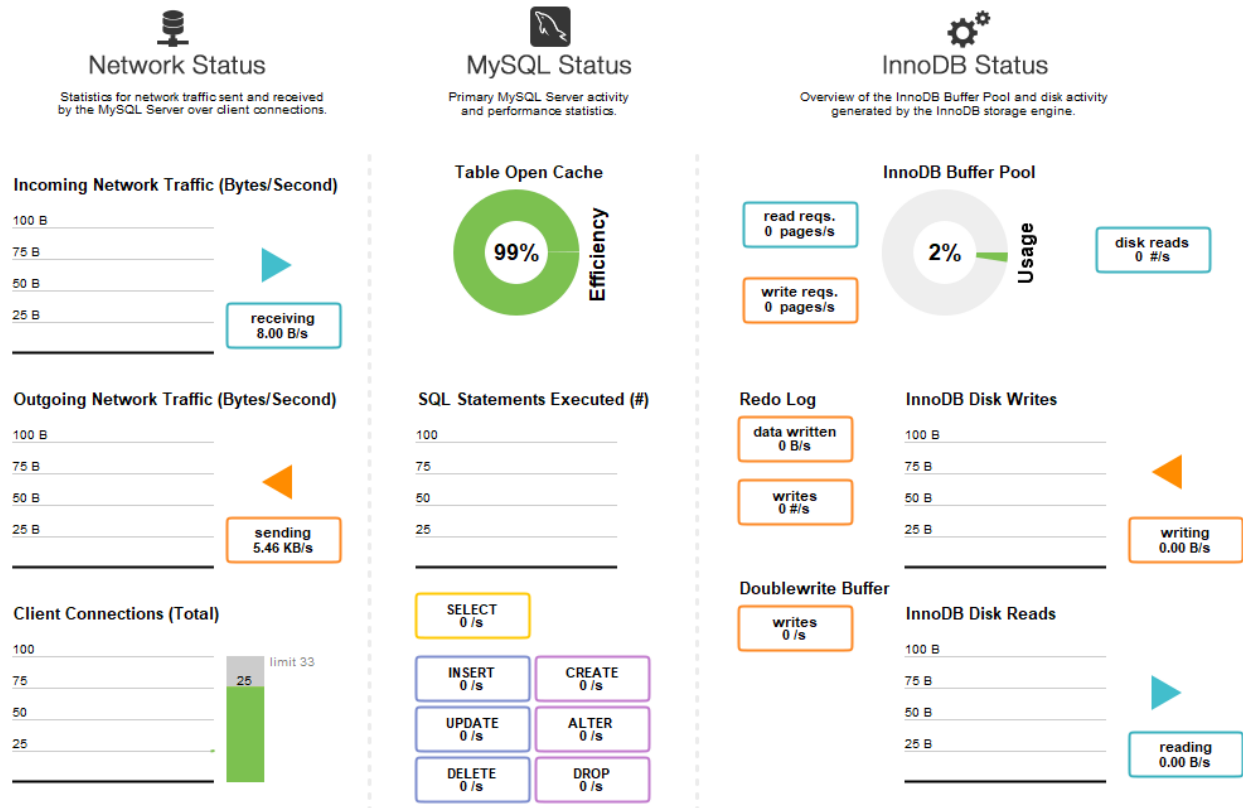


Figure 6 - Database Performance Diagram

## 6.4 System Design

### 6.4.1 Use-Cases

The use case diagram is located in the Requirements section - [Here](#)

### 6.4.1 Backend/Classes Design

Classes diagram UML

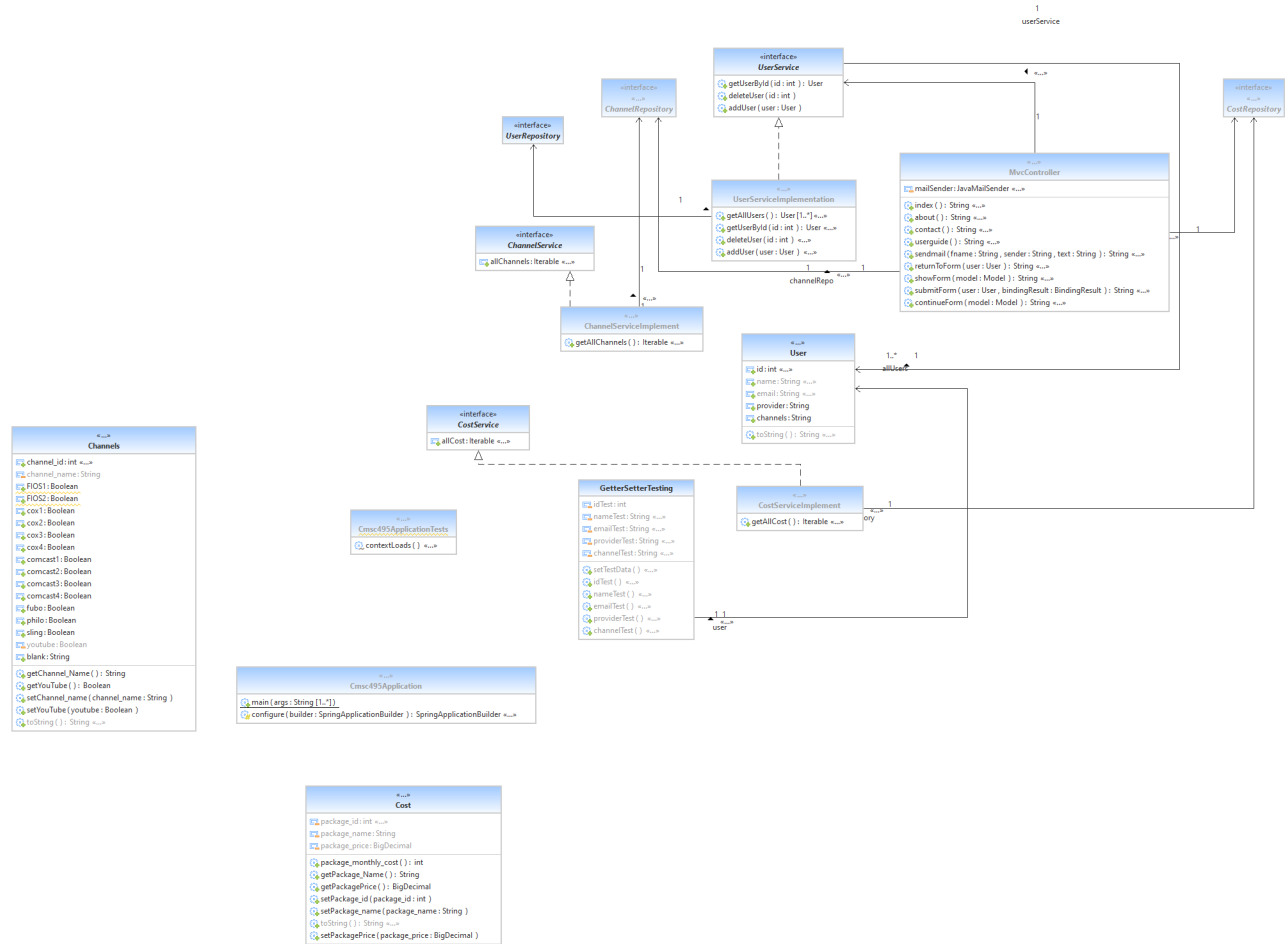


Figure 7 - Classes Diagram

### 6.4.2 Database Design

The database has a very simple design consisting of only three tables which do not reference each other. In this version the tables will only need to be used to reference data when requested from the MVC. In future version's it would be nice to incorporate a more robust database that will be designed so that all tables can be tied together and structured. To accomplish this most or all tables will require foreign key assignment and of course more robust data/tables will be required as well.

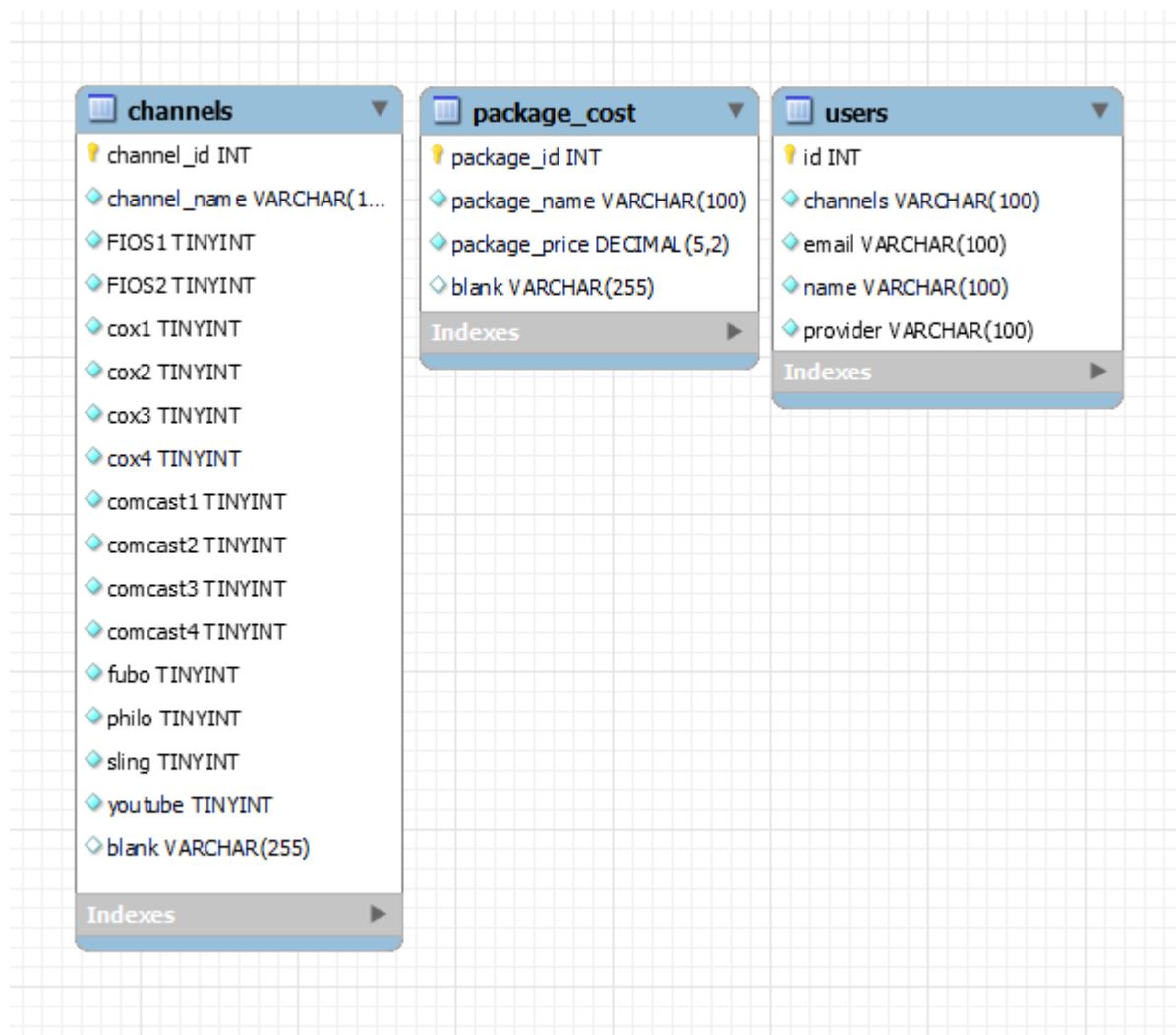


Figure 8 - Database Diagram (No Relations)

## 7 Version History

### 7.1 Introduction

This project did not use any software-based product for version control, but team is aware of the multiple options available. Instead, version control was done in combination with the agile scrum scheduling within Jira. If a version was scheduled to occur in Jira and did not those changes were pushed to the next scheduled version release date and that version release date was updated. Stakeholder comments will be included if given.

## 7.2 Version History

### 7.2.1 Version 0.0

- 0.1: Design front end.
- 0.2: Configure IDE
- 0.3: Draft database
- 0.4: Plan Classes

### 7.2.2 Version 1.0 (No Comments)

- 1.1: Design Application classes
- 1.2: Build pom dependencies based on class drafts
- 1.3: Test run as spring boot on hello world
- 1.4: draft user input tests
- 1.5: convert front end design to JSP using JSTL and HTML

### 7.2.3 Version 2.0 (No Comments)

- 2.1: Build database statements and run them (create database tables)
- 2.2: Configure application.properties file to allow AWS connection
- 2.5: Write code for the MVC which will be MvcController class
- 2.4: Write user repository, userservice, userserviceimplementation classes
- 2.5: Add all functions to MVC that will communicate from index.jsp to the database
- 2.6: Write about, contact, results, success and userguid pages.

### 7.2.4 Version 3.0 (No Comments)

- 3.1: Write Cost and Channel classes along with supporting services.
- 3.2: Write JSTL code for results.jsp that will populate a report based on the user's current cable subscription
- 3.3: Remove requirement for users zipcode entry since only three cable providers are being used.
- 3.4: Add javascript print function to allow user to print report.
- 3.5: Edit contact.jsp code to remove last name and include user's email.
- 3.6: Write controller into MVC for send email using gmail as the mail server.
- 3.7: Write mailsent.jsp to inform user that their email was sent without error.
- 3.8: Begin documenting/commenting all classes.

### 7.2.5 Version 4.0 (Final)

- 4.1: Write JavaDoc Comments and JSP comments to classes and pages.
- 4.2: Run all final testing.
- 4.3: Conduct final tweaks/review and tidy up code.

## 8 Conclusions

### 8.1 Introduction

This section of the final report will include expectations set initially, and where the final application landed in regard to those first expectations and what changes can be made to better the application.

The Cord Cutter Web Application met all the expectations initially planned except to take in the users Zip code and users Location based on that zip code. The decision to eliminate location was made due to there only being three cable TV providers listed in the app.

Improvements:

- Add more cable TV providers and packages which would require the need for a user location to determine what providers service what locations.
- To expand on the last improvement a cascading drop down should also be added. This for example would start with location and base the next dropdowns contents on that location, which would be location specific cable providers then the next dropdown would be packages based on the selected provider. This would eliminate any confusion from the user and make the app cleaner.
- Not using JSP, its not that anything specifically is wrong with using JSP its just the capabilities of JSP's is limited by what is available in JSTL.
- Make the reports page more dynamic, allow the user to select different streaming packages based on their current TV channels and the only add up the selected packages in the price comparison below.

## Appendix

### Appendix: References

The following table summarizes the documents referenced in this document.

Table 13 - Documentation

Document Name and Version	Description	Location
TestNG Documentation	Documentation for TestNG on installation and use.	<a href="https://testng.org/doc/documentation-main.html">https://testng.org/doc/documentation-main.html</a>
Jira Documentation	Documentation for Jira on setup and use.	<a href="https://confluence.atlassian.com/jira/jira-documentation-1556.html">https://confluence.atlassian.com/jira/jira-documentation-1556.html</a>

IEEE 829-2008-IEEE Standard for Software and System Test Documentation	Industry standards for software testing documentation	<a href="https://standards.ieee.org/standard/829-2008.html">https://standards.ieee.org/standard/829-2008.html</a>
--	---	---

## Appendix: Key Terms

The following table provides definitions for terms relevant to this document.

**Table 14 - Terms And Definitions**

Term	Definition
End-to-End Testing	Software testing method that involves testing an application's workflow from beginning to end.
Functional Testing	Software testing method that involves testing that a piece of software within an application.
Unit Testing	Software testing method that involves testing each unit of an application.
Compatibility Testing	Software testing method that involves testing an application works properly across different browsers, databases, operating systems, mobile devices etc.
UI Testing	A software testing method that involves testing the visual elements of an application.
Model View Controller (MVC)	a software design pattern commonly used for developing user interfaces that divide the related program logic into three interconnected elements. This is done to separate internal representations of information from the ways information is presented to and accepted from the user
Jakarta Server Pages (JSP)	formerly JavaServer Pages is a collection of technologies that helps software developers create dynamically generated web pages based on HTML, XML, SOAP, or other document types. Released in 1999 by Sun Microsystems,[1] JSP is





	<p>similar to PHP and ASP, but uses the Java programming language.</p> <p>To deploy and run Jakarta Server Pages, a compatible web server with a servlet container, such as Apache Tomcat or Jetty, is required.</p>
JavaServer Pages Standard Tag Library (JSTL)	<p>JavaServer Pages Standard Tag Library (JSTL) encapsulates as simple tags the core functionality common to many Web applications. JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags. It also provides a framework for integrating existing custom tags with JSTL tags.</p> <p>The JSTL 1.2 Maintenance Release aligns with the Unified Expression Language (EL) that is being delivered as part of the JavaServer Pages (JSP) 2.1 specification. Thanks to the Unified EL, JSTL tags, such as the JSTL iteration tags, can now be used with JavaServer Faces components in an intuitive way.</p> <p>JSTL 1.2 is part of the Java EE 5 platform.</p>

## Appendix: File Attachments

The following table includes outside file attachments.

**Table 15 - Attachments**

File Name	File
Timeline	 TeamTwoTimeline.xlsx lsx

<b>User Guide</b>	 UserGuide.docx
-------------------	---

## Appendix: Test Report

Below is the generated report from the 'TestNG' plugin.

### Default test

Tests passed/Failed/Skipped:	5/0/0
Started on:	Wed Nov 24 17:43:23 EST 2021
Total time:	0 seconds (25 ms)
Included groups:	
Excluded groups:	

*(Hover the method name to see the test class name)*

PASSED TESTS			
Test method	Exception	Time (seconds)	Instance
<b>channelTest</b> Test class: com.project.CMSC495.GetterSetterTesting Test method: Tests the getter and setter of channels		0	com.project.CMSC495.GetterSetterTesting@41294f8
<b>emailTest</b> Test class: com.project.CMSC495.GetterSetterTesting Test method: Tests the getter and setter of email		0	com.project.CMSC495.GetterSetterTesting@41294f8
<b>idTest</b> Test class: com.project.CMSC495.GetterSetterTesting Test method: Tests the getter and setter of id		0	com.project.CMSC495.GetterSetterTesting@41294f8
<b>nameTest</b> Test class: com.project.CMSC495.GetterSetterTesting Test method: Tests the getter and setter of name		0	com.project.CMSC495.GetterSetterTesting@41294f8
<b>providerTest</b> Test class: com.project.CMSC495.GetterSetterTesting Test method: Tests the getter and setter of provider		0	com.project.CMSC495.GetterSetterTesting@41294f8

Figure 9 - Test Report

## UI Tests

Feature	Expected Behavior	Actual Behavior	Pass/fail
"Print Page" button	Will prompt user with the option to print the results based on given selections.	Prompts users with the option to print the results based on given selections.	Pass
About link on website	Redirects user to /about	Redirects user to /about	Pass
User Guide link on website	Redirects user to /userguide	Redirects user to /userguide	Pass
Contact link on website	Redirects user to /contact	Redirects user to /contact	Pass
Submit Button on /home	Does not allow user to proceed without fields filled and will redirect to /submit to verify options otherwise.	Does not allow user to proceed without fields filled and will redirect to /submit to verify options otherwise.	Pass
Reset Button on /home	Resets currently selected options to default	Resets currently selected options to default	Pass
Back Button on /submit	Returns user to /home	Returns user to /home	Pass
Continue button on /submit	Redirects user to /results	Redirects user to /results	Pass
Service Combo box	Combo box dropdown shows all service options	Combo box dropdown shows all service options	Pass
Package Combo box	Combo box dropdown shows all package options	Combo box dropdown shows all package options	Pass

### Compatibility Tests

Feature	Expected Behavior	Actual Behavior	Pass/fail
Firefox end-to-end test	Execute an end-to-end test with no errors	Execute an end-to-end test with no errors	Pass
Google Chrome end-to-end test	Execute an end-to-end test with no errors	Execute an end-to-end test with no errors	Pass
Microsoft Edge end-to-end test	Execute an end-to-end test with no errors	Execute an end-to-end test with no errors	Pass

### Unit Tests

Feature	Expected Behavior	Actual Behavior	Pass/fail
Getter/Setter of integer id	Output: "idTest"	Output: "idTest"	Pass
Getter/Setter of String name	Output: "nameTest"	Output: "nameTest"	Pass
Getter/Setter of String email	Output: "emailTest"	Output: "emailTest"	Pass
Getter/Setter of String provider	Output: "providerTest"	Output: "providerTest"	Pass
Getter/Setter of String channel	Output: "channelTest"	Output: "channelTest"	Pass

### Functional Tests

Feature	Expected Behavior	Actual Behavior	Pass/fail
Email validation	Prompt user if invalid email format	Prompts user if invalid email format	Pass
Subscription results	Report shows which subscriptions are already owned based on selection	Report shows which subscriptions are already owned based on selection	Pass
Savings results	Results show potential savings for each subscription service	Results show potential savings for each subscription service	Pass
Date & Time	Date and time displayed on webpage are accurate at the time of load	Date and time displayed on webpage are accurate at the time of load	Pass

### End-to-End Tests

In order to test compatibility on the three most common browsers, end-to-end tests are performed on each browser for compatibility as well.