

CS 320 Course Project Final Report

for

PDF Share & Schedule

Prepared by

Group Name: R&B

|  |  |  |
| --- | --- | --- |
| Ben Mikalenko | 11589806 | Ben.mikailenko@wsu.edu |
| Roman Stolyarov | 11689068 | Roman.stolyarov@wsu.edu |

|  |  |
| --- | --- |
|  |  |
| Date: | Friday, October 25th 2019 |

Contents

Contents ii

1 Introduction 1

1.1 Project Overview 1

1.2 Definitions, Acronyms and Abbreviations 1

1.3 References and Acknowledgments 1

2 Design 2

2.1 System Modeling 2

2.2 Interface Design 2

3 Implementation 3

3.1 Development Environment 3

3.2 Task Distribution 3

3.3 Challenges 3

4 Testing 4

4.1 Testing Plan 4

4.2 Tests for Functional Requirements 4

4.3 Tests for Non-functional Requirements 4

4.4 Hardware and Software Requirements 4

5 Analysis 5

6 Conclusion 6

Appendix A - Group Log 7

# 

# Introduction

*<TO DO: Please provide a brief introduction to your project.>*

## Project Overview

PDF Share and Schedule is an online application that allows users to upload PDF documents to a centralized location. PDF documents are grouped, shown on the home pag and organized by user inputted date. Through the upper menu GUIs, the user can add, delete, and edit the groups of PDF documents.

## Definitions, Acronyms and Abbreviations

Chord Chart – A form of musical document that lists a songs lyrics along with the chords played at each section

Date (in regards to the webpage) – An entry to the main webpage that has a real calendar date as a name and has zero or more PDFs attached and avalible for download.

GUI – General User Interface

PDF – Portable Document Format

Main Page – The div with the id of MainPage in the index.html where all the user inputted content is stored

## References and Acknowledgments

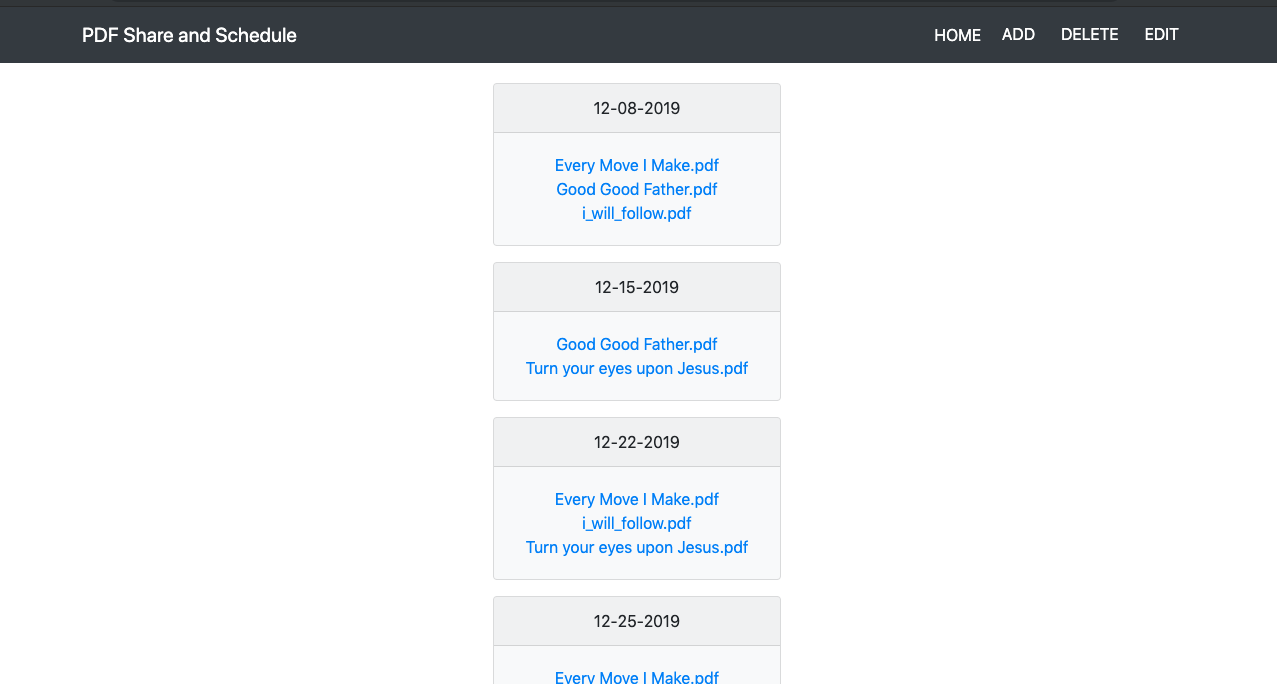
NO REFERENCES YET (USE IEEE CITATION IF WRITING A REFERENCE)

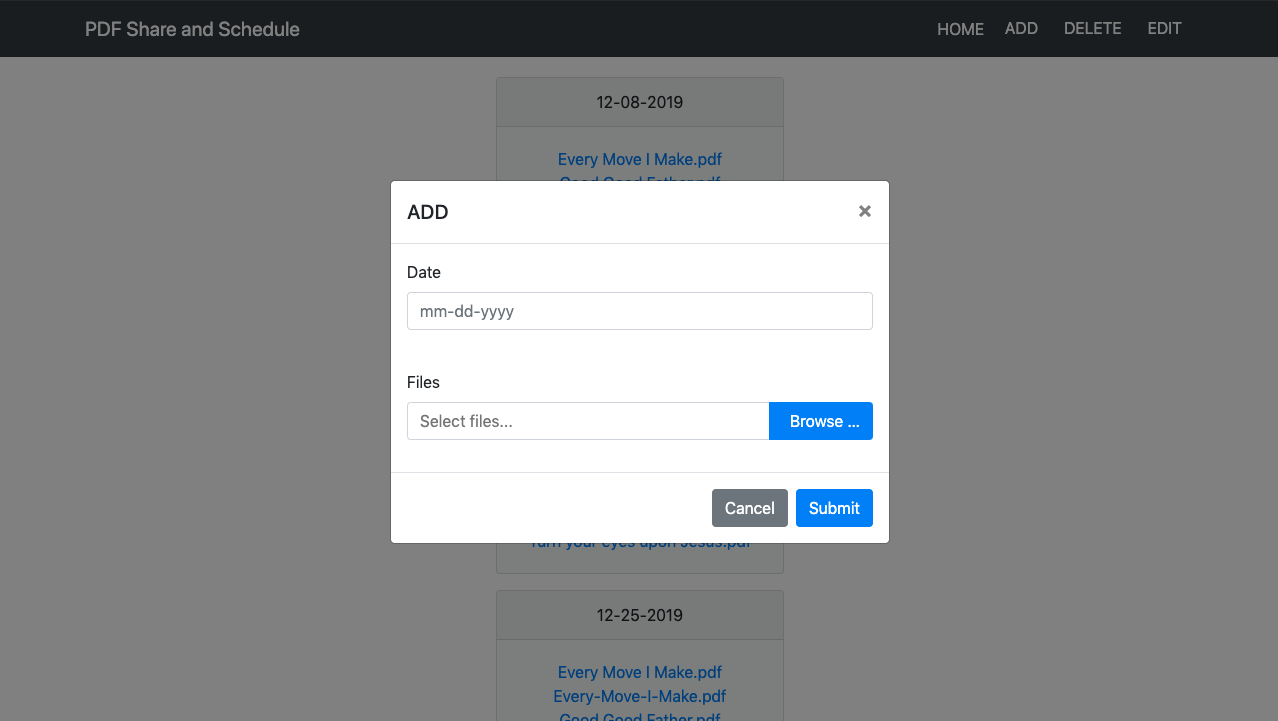
# Design

## System Modeling

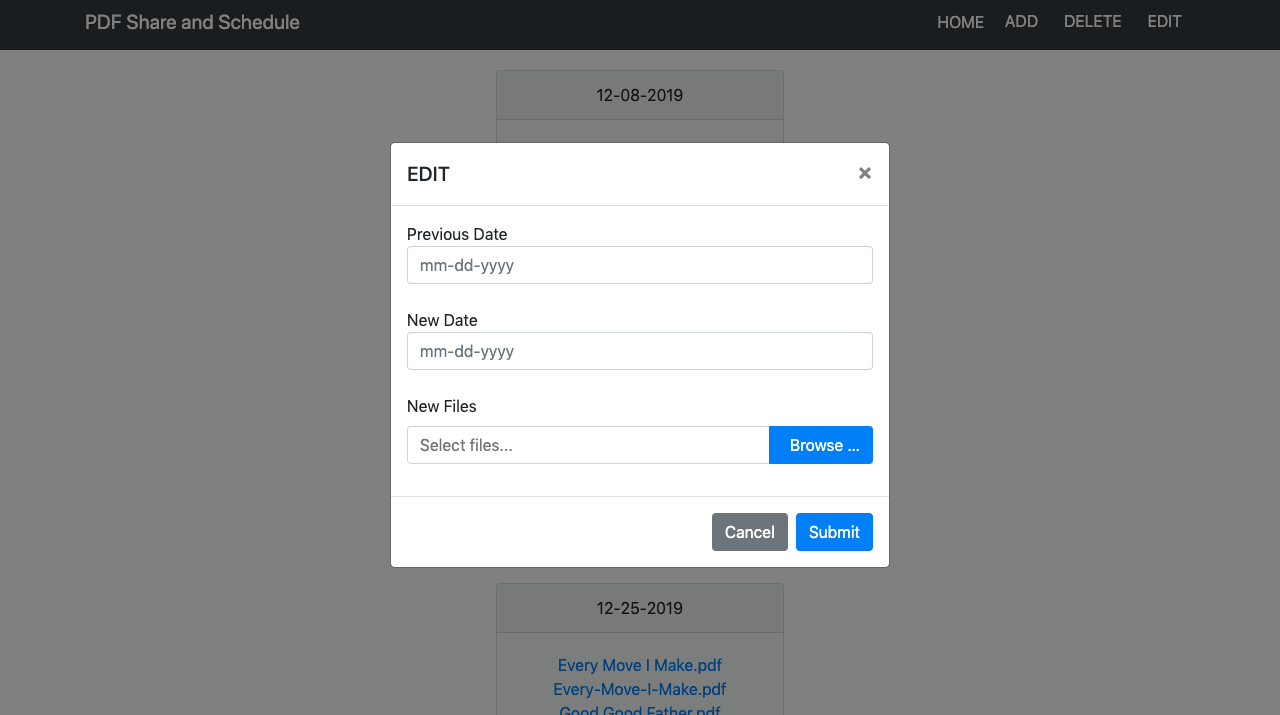
Our inplementaion strictly follows the design document in milestone 2. But note, in the design documents “structural modeling” section, “entry” and “main page” are reffered to the divs with id’s of dates and the div of id “MainPage” accordingly.

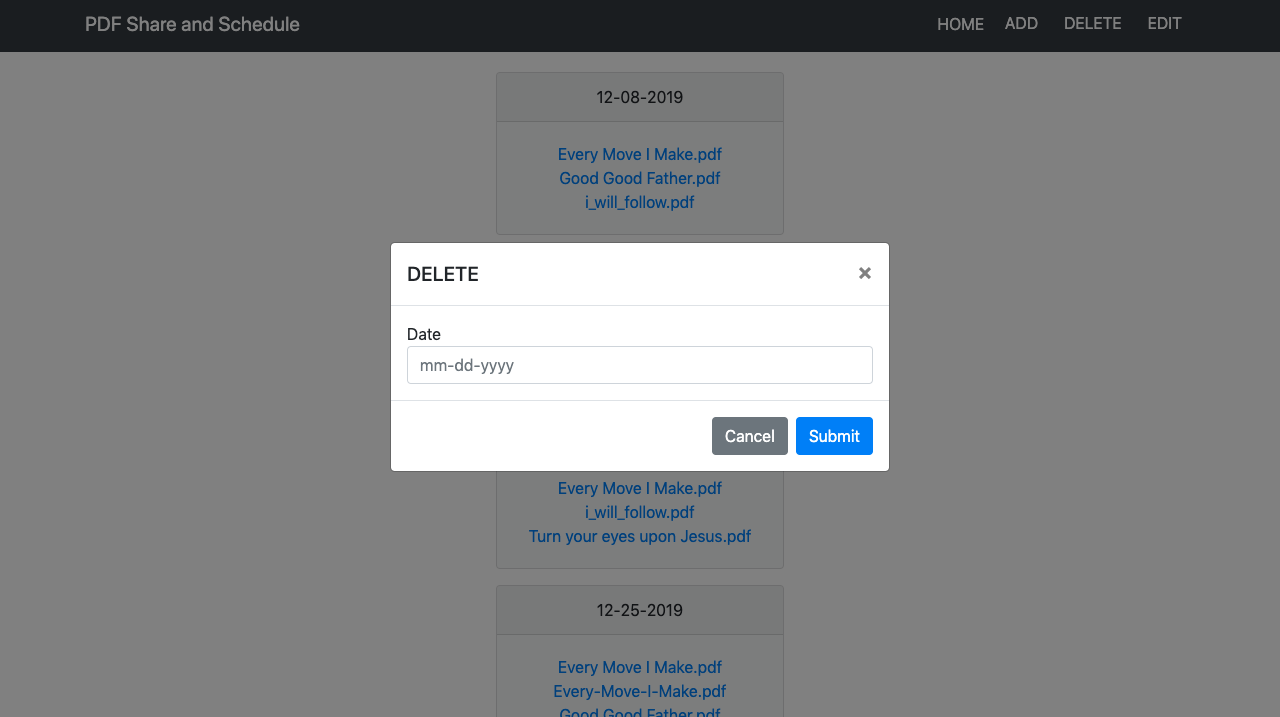
## Interface Design

Main page

Add GUI

Edit GUI



Delete GUI

# Implementation

## Development Environment

HTML and Javascript was used for the implementation. Plenty of UI frameworks like bootstrap, bootstrap-datepicker, and bootstrap-fileinput were used inside the of the HTML. Node js + express was the main web application framework used for the backend.

UI Framworks used:

Bootstrap – Main layout and look of the page, buttons, and main page entries (bootstrap cards)

Bootstrap-datepicker – Calendar pop-pop in the GUIs date input field

Bootstrap-fileinput – Allowed for multiple file input from only one input field

Javascript framworks used:

Node js – Runtime enviornment

Express js – Application framework

Express-fileupload – Allowed express to easily put uploaded files into the req.body.files

Mv – Moved uploaded files to local directory

Fs – Allowed the use of the local directory

JsDom/JQuery – Allowed the manipulation of the HTML files

Del – Deleted entire folders quickly

Multer – Allowed us to use multipart form data

## Task Distribution

Implentation was done in pairs and individually. Most of the work was done on one computer to save time pushing to github. But code implementation was being done by both team members equally. All of the design, research, and implementation was done together.

## Challenges

The most challenging part was to actually upload the files to the application directory along with the date (text) input. Multipart data was being inputted from the forms, but express-fileupload only received the files and not the date input. We used multer to receive the text input and express-fileupload to receive the files.

# Testing

## Testing Plan

Testing will begin once the backend implementation is complete. Estimated time of testing is the last week of November 2019. Things to be tested:

1. The application redirects with no errors or crashes to the home page when the submit button is hit and there are no files inputted.
2. The application deletes all the selected files from the applications “files” directory from the delete GUI.
3. All entries to the main page are organized by date at all times.
4. 1+ files can be inputted and uploaded successfully.
5. The site must load within 3 seconds
6. A entry to the main page just take less then 5 seconds

## Tests for Functional Requirements

When no input files, redirect in app.post(/upload and /edit) – PASSED, correctly redirected

When delete is executed, app.post(/delete) removes files – PASSED, correctly deleted with ‘Del’

All entries are ordanised in app.post(/upload and /edit) – PASSED, JQuery organised correctly

1+ files can be inputted and uploaded in app.post(/upload and /edit) – PASSED, boostrap-fileinput worked correctly

## Tests for Non-functional Requirements

3 second website load – PASSED, the website loaded less than 1 second.

5 second entry to main page – PASSED, the entry added less then 1 second. Note, the page might need refreshing to show the new entries.

## Hardware and Software Requirements

The hardware required is any device that can use a modern web browser. Since the javascript and frameworks are relativley new and updated, older systems cannot use the application.

The software required is any new web browser (the most updated version of chrome, firefox, etc).

# Analysis

Milestone 1 – SRS – Both team members spent 2 hours, Tuesdays and Thursdays, until project due date, working on the document.

Milestone 2 – Design Document – Both team members spent 4 hours together working on the design document.

Milestone 3 – Software, demo, final report – Team members worked together on the software and final project around 5 hours per week. This proved the most challenging because of the actual software implementation.

# Conclusion

PDF Share and Schedule proved a worthy challenge. Every step of the way was another roadblock leading to hours of documentation. Managing the project versions with GitHub took out more time learning than saving. Simple solutions like ‘refreshing to see the current state of the webpage’ were the hardest to realize. But in the end, the knowledge gained from this hands-on experience taught us so much more. Not one second went to waste.

We learned about an SRS document. The importance of an SRS document is everyone was on the same page, all the time. Next time we implement an SRS document, we would do more research on the actual software frameworks, languages, and implementations. Just a little bit more knowledge before hand about the software would have helped plenty with ambiguity in the document and saved time.

We learned about a design document. It’s good for showing the overall structure of the program. Next time we implement the design document we would have added much more detail. A lot more information could have been added to the designs. More information could have proved useful.

We learned about implementing a web application. PDF Share and Schedule had lots of external help (bootstrap, multer, express-fileupload) that sped up the implementaiton process. A good chunk of the work was just reading documentation. We think that after this project, we have become more accoustomed to reading documentation and integrating other pre-written, open source code into our work. Which is important because the world of programming is constantly changing. Syntax, frameworks, and add-ons along with it. More skill in documentation understanding will always be beneficial.

Appendix A - Group Log

Group members met Tuesdays and Thursdays, 1:20pm to around 2:50pm every week for the entire lifetime of the project. Communication proved effective through github and text messaging.