CMSC 447: Software Engineering I

**UMBC ParkingPal**

**System Design Document**

Customer:

Katherine Gibson

Team: UMBC Construction Workers

Abbie Minor

Constantin Koehler

Braxton Dubin

Naomi Schumacher

Sarah Kirby

UMBC ParkingPal

System Design Document

**Table of Contents**

[**1. Introduction**](#_z65ny3trzxfc)[**3**](#_z65ny3trzxfc)

[1.1 Purpose of This Document](#_dsotppfw5usn) [3](#_dsotppfw5usn)

[1.2 References](#_tz3yhjtj0j6e) [3](#_tz3yhjtj0j6e)

[**2. System Architecture**](#_ohj2msgj8fk1)[**3**](#_ohj2msgj8fk1)

[2.1 Architectural Design](#_1553umjo25wg) [3](#_1553umjo25wg)

[2.2 Decomposition Description](#_y4vgsasoxfhv) [4](#_y4vgsasoxfhv)

[**3. Persistent Data Design**](#_8cmkup4bvuvs)[**5**](#_8cmkup4bvuvs)

[3.1 Database Descriptions](#_9qt86zn8adto) [5](#_9qt86zn8adto)

[3.2 File Descriptions](#_6d9itn2bjiez) [6](#_6d9itn2bjiez)

[**4. Requirements Matrix**](#_a0yug7acox2r)[**6**](#_a0yug7acox2r)

[**Appendix A – Agreement Between Customer and Contractor**](#_i4g8jwia1hj2)[**8**](#_i4g8jwia1hj2)

[**Appendix B – Team Review Sign-off**](#_bw3t24koe7r9)[**9**](#_bw3t24koe7r9)

[**Appendix C – Document Contributions**](#_sa34kuiwwqby)[**10**](#_sa34kuiwwqby)

# 

# 1. Introduction

## 1.1 Purpose of This Document

The purpose of this document is to clearly define the webapp at the architectural level, including but not limited to descriptions of architectural design, databases, and files used by the system. This document is important to the customer to model what is going on behind the scenes to make this webapp run smoothly. This document also helps by laying down the foundation of UMBC ParkingPal for the development team.

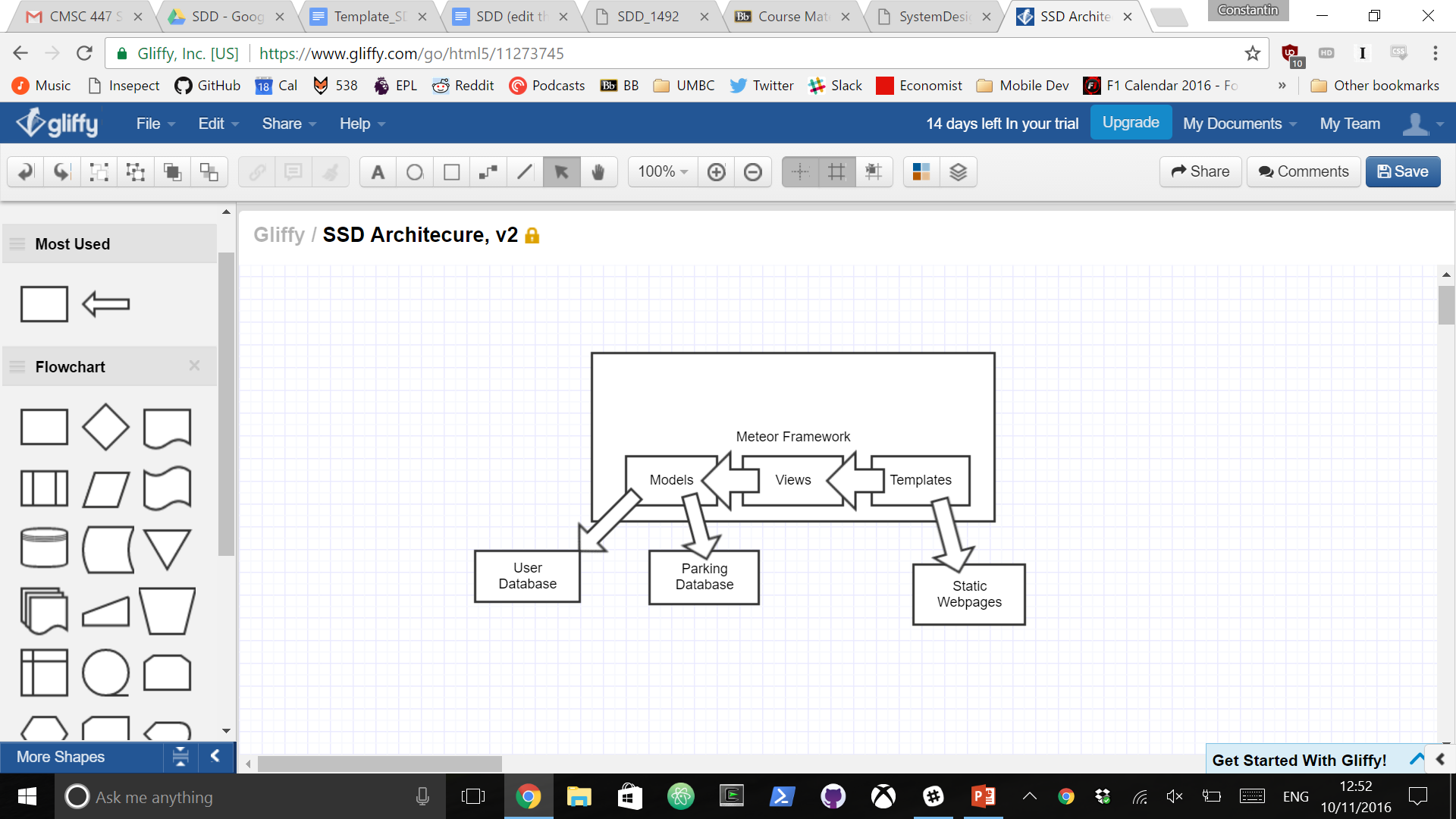
## 1.2 References

UMBC ParkingPal System Requirements Specification

UMBC ParkingPal User Interface Design Document

# 2. System Architecture

## 2.1 Architectural Design

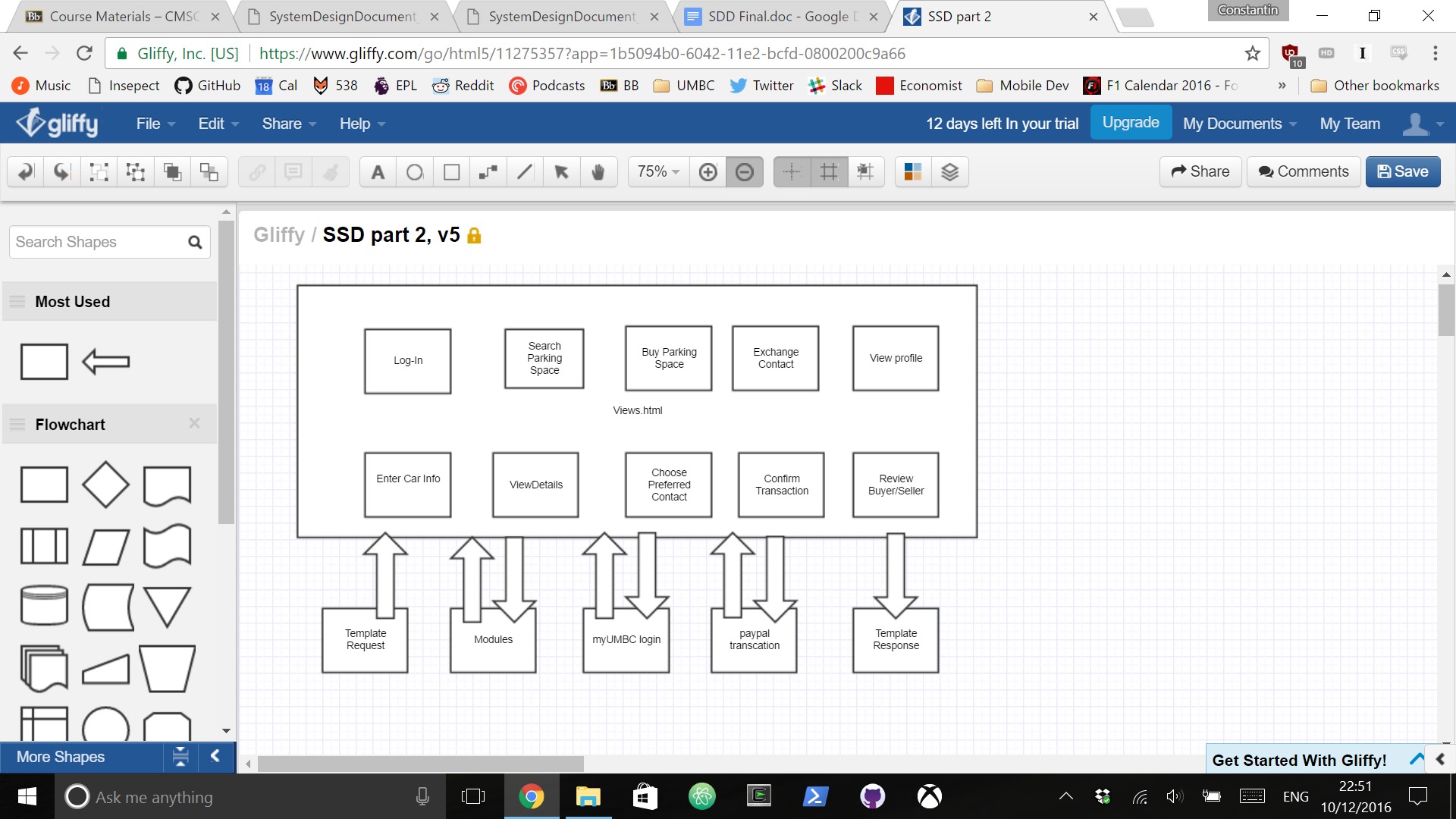


The UMBC Parking Pal system will be using the Meteor framework. The Meteor framework follows the MVC (Model, View, Controllers) architecture, which means it has Models, Views and Templates. The Model layer refers to the tables in our database. These tables will be implemented as a user database and a parking lot database. The user database will contain all the data about our users including their driver information, email, information about their cars, and parking pass status. The parking database will contain information about each of the lots on campus and the type of students that may park there given their parking pass type.

As the template is the webpage itself, we will be using HTML to define our forms, buttons, dropdowns and other UI elements. We will use CSS with Bootstrap to make our templates responsive on desktop and mobile devices as well as making them more aesthetically pleasing for the user. These templates are described in greater detail in our UID (User-interface Design) Document.

Views in our application include searching, reviewing other users, and buying or selling parking spaces. These views are facilitated by our databases that contain information about our users and parking lots they wish to occupy.

## 2.2 Decomposition Description



Each view above will have its own HTML file detailing its layout, requests and responses. The Login view will ask the user to login using their myUMBC credentials. If the corresponding e-mail is not already in our user database, they will be asked to fill out information about their parking pass status as well as information about their car. These fields are specified in the image labeled User Information below. The ability to search for specific parking spaces is based on the user’s parking pass settings from their profile. The information the buyer can see regarding parking information can be seen in the Parking Information diagram below.

The View Details view gives the buyer a rough location of the parking spot for sale as well as information about the seller. The Exchange Contact view instructs the buyer to contact the seller over text-message or phone call and shares the seller’s phone number to the buyer. From there, the two users are able to communicate over text-message or call and are able to confirm the transaction in the Confirm Transaction view. The monetary transaction occurs over Paypal, where the money is transferred after both parties agree to the transaction. After the transaction, the buyer can view the seller’s profile and leave feedback on their profile.

Within the Rate view, the buyer is able to leave feedback about the seller which is tied to their university e-mail. If a user consistently receives low ratings after completing a transaction, we will be able to ban the user from using our webapp any further. However, we don’t anticipate this becoming a problem as there is no anonymity to hide behind.

User Information Parking Information

# 3. **Persistent Data Design**

## 3.1 Database Descriptions

The database that will be used will be MongoDB and the data will be stored in JSON documents. MongoDB uses collections instead of tables, and each entry in the collection is called a document. MongoDB creates a unique key for each document, stored in the ObjectId field. This is the primary key for every collection.

There will be two collections in our database. The first is the Users collection, which stores User objects containing the user’s UMBC email, contact info, car info, permit type and active listings or purchases. The active listing is a foreign key, and must hold an ObjectId for a listing in the Listings collection. The second collection is the Listings collection. This stores all active listings as a Listing object containing the lot number, time available, price, seller ID and buyer ID. The seller ID and buyer ID stored for each listing are foreign keys. The ID’s stored in these fields will correspond to a user’s ObjectId, so they can be used to lookup the buyer and seller’s information in the Users collection. A listing must always at least have a SellerId stored.

Refer to the following tables for a description of the objects and fields stored in each collection. Note that JSON does not limit string sizes, so the string sizes are based on the limits enforced on data entry in the user interface.

**Users Collection:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Object** | **Field** | **Type** | **Size** | **Description** |
| User | ObjectId | String | 12 bytes | Unique identifier created by MongoDB |
| Email | String | 30 bytes | User’s UMBC email address |
| Phone | String | 10 bytes | User’s phone number, numbers only |
| Permit | String | 1 byte | User’s permit type, single character |
| CarInfo | Object | 68 bytes | Object that holds user’s car info |
| Rating | Double | 8 bytes | Average rating for the user, between 0 and 5. Initialized to -1. |
| CarInfo | Make | String | 20 bytes | Car make |
| Model | String | 20 bytes | Car model |
| Color | String | 20 bytes | Car color |
| LicensePlate | String | 8 bytes | User’s license plate number, including hyphens |

**Listing Collection:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Object** | **Field** | **Type** | **Size** | **Description** |
| Listing | ObjectId | String | 12 byte | Unique identifier created by MongoDB |
| SellerId | String | 12 byte | Seller’s ObjectId |
| BuyerId | String | 12 byte | Buyer’s ObjectId, initialized to -1. |
| Price | Integer | 4 bytes | Price of the spot. |
| SpotTime | Object | 8 bytes | Object storing hour and minute when the spot will be available. |
| Lot | Integer | 4 bytes | The lot number of the spot |
| SpotTime | Hour | Integer | 4 bytes | Hour the spot will be available, stored as military time (0-24) |
| Minute | Integer | 4 bytes | Minute the spot will be available (0-59) |

## 3.2 File Descriptions

There are several files used in the implementation of UMBC ParkingPal. First, we store our data in dynamic JSON documents using MongoDB. The JSON data and file format is covered in the Database Description. There are also several static files used in our implementation.

The static files that compose our front-end include HTML, CSS, and JavaScript files. The HTML files contain templates for web pages, including layout code as well as some logic using Meteor’s Spacebars. The HTML files are compiled with Meteor’s Spacebars compiler. The CSS files contain styling information for those HTML pages. The JavaScript files are necessary to use the Meteor framework. They utilize the HTML templates and allow us to display dynamic data from our database collections.

Our backend consists of separate JavaScript files which control access to our database. The JavaScript files contain functions which retrieve JSON documents from our database and format them so they can be used by the front-end. JavaScript functions will also handle input from the front-end forms, and will perform data validation before making updates to the database.

# 4. Requirements Matrix

Please refer to the System Requirements Specification for details regarding corresponding use cases.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case # | Name | Attributes Involved | System Involved |
| 1 | Register | User: Email  User: PermitType  User: CarMake  User: CarModel  User: CarColor  User: LicensePlateNumber  User: PhoneNumber | User Database |
| 2 | Login | User: Email | User Database  myUMBC Login |
| 3 | User lists their parking spot | Listing: LotNumber  Listing: Price  Listing: TimeLeaving  Listing: Lister  Listing: Number  User: Email  User: PermitType | Parking Spot Database  User Database |
| 4 | User searches for parking spot | User: PermitType  Listing: LotNumber  Listing: Price  Listing: TimeLeaving | Parking Spot Database  User Database |
| 5 | User purchases parking spot | Listing: Price  Listing: Lister | Parking Spot Database  User Database  Paypal |
| 6 | Buyer and Seller exchange parking spots | Listing: Lister  Listing: Number  User: Email | Parking Spot Database  User Database |
| 7 | Buyer cancels their purchase | Listing: Number  Listing: Lister | Parking Spot Database |
| 8 | Seller cancels their listing | Listing: Number | Parking Spot Database |

# 

# 

# Appendix A – Agreement Between Customer and Contractor

By signing this document all members of the team agree to complete the project with the design specified.

If future changes need to be made to the design, the team members agree to consult with the customer and vice versa. If an agreement is made, it must be made in writing and the design documents must be updated with the changes.

Abbie Minor \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Constantin Koehler \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Braxton Dubin \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Naomi Schumacher \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Sarah Kirby \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Katherine Gibson \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

(Customer)

Customer Comments:

# Appendix B – Team Review Sign-off

All members of the team have reviewed this document and agree on its content and format. They agree to follow the design specified for the development of UMBC ParkingPal. They agree to discuss possible areas for change with other team members and follow the guidelines for making changes to the design. The comment areas below are to be used to state any minor points regarding the document that members may not agree with.

Abbie Minor \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Comments:

Constantin Koehler \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Comments:

Braxton Dubin \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Comments:

Naomi Schumacher \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Comments:

Sarah Kirby \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_

Comments:

# Appendix C – Document Contributions

This section identifies how each member contributed to the creation of this document. The percentages listed are an estimate of the percentage of work each person contributed.

Abbie Minor: 25%

Contribution: Table of Contents, Requirements Matrix

Constantin Koehler: 25%

Contribution: System Architecture

Braxton Dubin: 0%

Contribution:

Naomi Schumacher: 25%

Contribution: Appendices, proofreading

Sarah Kirby: 25%

Contribution: Persistent Data Design, File Descriptions, Document template