Software Design Document (SDD) Template

Software design is a process by which the software requirements are translated into a representation of software components, interfaces, and data necessary for the implementation phase. The SDD shows how the software system will be structured to satisfy the requirements. It is the primary reference for code development and, therefore, it must contain all the information required by a programmer to write code. The SDD is performed in two stages. The first is a preliminary design in which the overall system architecture and data architecture is defined. In the second stage, i.e. the detailed design stage, more detailed data structures are defined and algorithms are developed for the defined architecture.

This template is an annotated outline for a software design document adapted from the IEEE Recommended Practice for Software Design Descriptions. The IEEE Recommended Practice for Software Design Descriptions have been reduced in order to simplify this assignment while still retaining the main components and providing a general idea of a project definition report. For your own information, please refer to [IEEE Std 1016­1998](http://www.cs.concordia.ca/~ormandj/comp354/2003/Project/ieee-SDD.pdf)[[1]](#footnote-1) for the full IEEE

Recommended Practice for Software Design Descriptions.

ChromaticCanvasInc.

**ChromaticCanvas**

Software Design Document

Name (s):

ROSIU MARIUS

QASEM ODAI

ABDALLA ABDELKARIM

Lab Section: Workstation:

Date: 21/03/2021

**TABLE OF CONTENTS**

[1. INTRODUCTION 2](#_Toc4987)

[1.1 Purpose 2](#_Toc4988)

[1.2 Scope 2](#_Toc4989)

[1.3 Overview 2](#_Toc4990)

[1.4 Reference Material 2](#_Toc4991)

[1.5 Definitions and Acronyms 2](#_Toc4992)

[2. SYSTEM OVERVIEW 2](#_Toc4993)

[3. SYSTEM ARCHITECTURE 2](#_Toc4994)

[3.1 Architectural Design 2](#_Toc4995)

[3.2 Decomposition Description 3](#_Toc4996)

[3.3 Design Rationale 3](#_Toc4997)

[4. DATA DESIGN 3](#_Toc4998)

[4.1 Data Description 3](#_Toc4999)

[4.2 Data Dictionary 3](#_Toc5000)

[5. COMPONENT DESIGN 3](#_Toc5001)

[6. HUMAN INTERFACE DESIGN 4](#_Toc5002)

[6.1 Overview of User Interface 4](#_Toc5003)

[6.2 Screen Images 4](#_Toc5004)

[6.3 Screen Objects and Actions 4](#_Toc5005)

[7. REQUIREMENTS MATRIX 4](#_Toc5006)

[8. APPENDICES 4](#_Toc5007)

# INTRODUCTION

## Purpose

Identify the purpose of this SDD and its intended audience. (e.g. “This software design document describes the architecture and system design of XX. ….”).

The SDD paper is intended for programmers, system administrators and database administrators to describe the architecture and design of the project.

## Scope

Provide a description and scope of the software and explain the goals, objectives and benefits of your project. This will provide the basis for the brief description of your product.

The “Chromatic Canvas” web application is a reliable and ingenious tool that offers art enthusiasts a way to greatly increase their collection, share their preferences and expand their own circle of influence by participating or hosting impressive art galleries. All of this will be provided for a small commission that fallows an affordable monetization plan.

Since this is a web application the software will always require Internet access. All system information is maintained in a database, which is located on a web-server. The application will also allow a special log in option for administrators that will come with additional perks.

## Overview

Provide an overview of this document and its organization.

The document is spread in 8 different chapters , each with numerous subchapters. Each chapter formulates concepts regarding different parts of the project and describes the system at the architecture level. This including subsystems and their services, data management, component design, which describes what each component does in a more systematic way, and the human interface design which will provide the functionality of the system from the user’s perspective.

## Reference Material

*This section is optional.*

List any documents, if any, which were used as sources of information for the test plan.

[1] IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended

Practice for Software Requirements Specifications”, June 25, 1998.

[2] GitHub: <https://github.com/Aodai/ArtClub>

[3] California art club provided example text and imagery for the web application: <https://www.californiaartclub.org/>

## Definitions and Acronyms

*This section is optional.*

Provide definitions of all terms, acronyms, and abbreviations that might exist to properly interpret the SDD. These definitions should be items used in the SDD that are most likely not known to the audience.

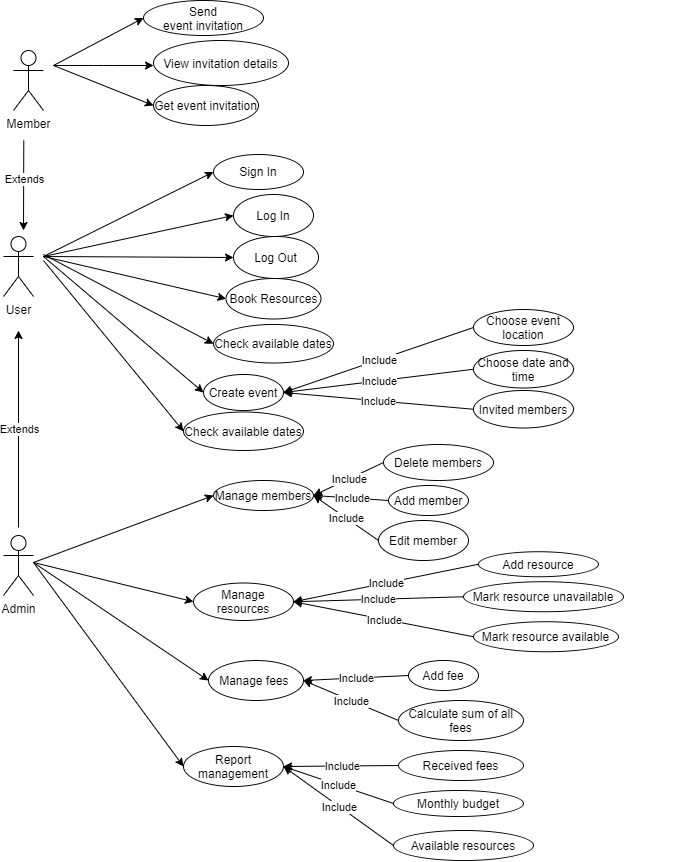
Inside of the document there are some terms that need to be explained beforehand:

|  |  |
| --- | --- |
| User | Person who interacts with the application |
| Member | User that has agreed to register and purchased the membership |
| Administrator | Staff member with special permissions and privileges who manages and controls the system |
| Resources | Object or places belonging to the art club that can be rented to members for events. |
| Fees | Costs required for creating an event (200 lei/day/resource if the user bought a membership or 400 lei/day/resource if the user did not buy a membership) or the monthly payment |
| Event | Meeting organized by a member/user/admin using the application. The meeting requires the use of different resources (provided by the art club) and funds(payment fees). |

# SYSTEM OVERVIEW

Give a general description of the functionality, context and design of your project. Provide any background information if necessary.

The system diagram presented below presents the capabilities of our art club web application. In this diagram we give a short overview of the user categories and what they can obtain from using our web application.



# SYSTEM ARCHITECTURE

## Architectural Design

Develop a modular program structure and explain the relationships between the modules to achieve the complete functionality of the system. This is a high level overview of how responsibilities of the system were partitioned and then assigned to subsystems. Identify each high level subsystem and the roles or responsibilities assigned to it. Describe how these subsystems collaborate with each other in order to achieve the desired functionality. Don’t go into too much detail about the individual subsystems. The main purpose is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together. Provide a diagram showing the major subsystems and data repositories and their interconnections. Describe the diagram if required.

--COMPLETE THIS—

## Decomposition Description

Provide a decomposition of the subsystems in the architectural design. Supplement with text as needed. You may choose to give a functional description or an object­oriented description. For a functional description, put top­level data flow diagram (DFD) and structural decomposition diagrams. For an OO description, put subsystem model, object diagrams, generalization hierarchy diagram(s) (if any), aggregation hierarchy diagram(s) (if any), interface specifications, and sequence diagrams here.

--COMPLETE HERE—

The Model View Controller (MVC) is an architectural pattern used in software engineering.

It is a way of creating client applications that leverages core features of the WPF platform and allows for simple unit testing of application functionality.

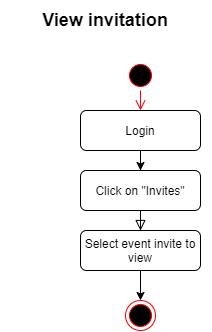
**Model**: Model represents the shape of the data. A class in C# is used to describe a model. Model objects store data retrieved from the database.

**View**: View in MVC is a user interface. View display model data to the user and also enables them to modify them. View in ASP.NET MVC is HTML, CSS, and some special syntax (Razor syntax) that makes it easy to communicate with the model and the controller.

**Controller**: The controller handles the user request. Typically, the user uses the view and raises an HTTP request, which will be handled by the controller. The controller processes the request and returns the appropriate view as a response.

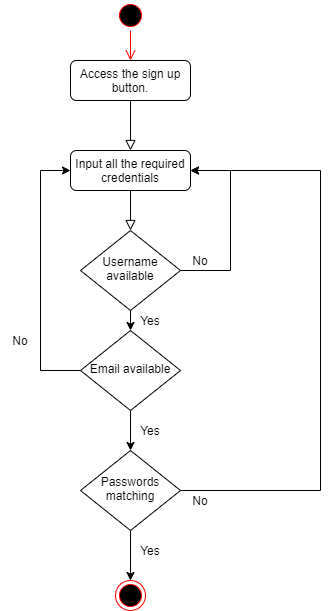
**1.View Invitation activity diagram**

Using our web application the user has the capability to view their invites by firstly executeing the Log In procedure then ending then accessing the invites button.

****

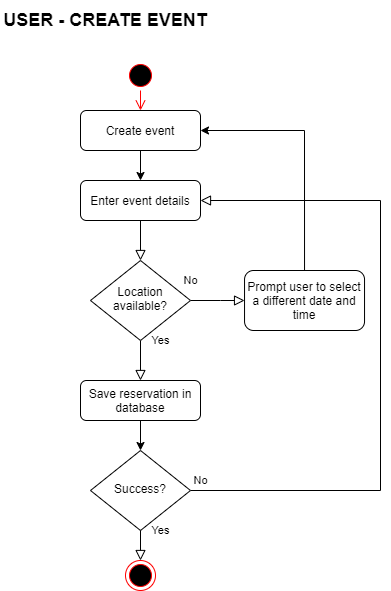
**2.Sign Up activity diagram**

Before accessing our web application the user must first create an account. This procedure is executed by firstly pressing the sign up button then inputing all their credential appropriately. If all of their credentials are correct, then the user will be able to create a new account, if not they will be forced to retry inserting their credentials.

****

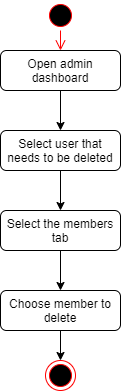
**3.Create Event activity diagram**

Using our web application the user has the option to create an event. To do so they must access the create an event button. Afterwards they will have to fill in the event details which will result in one of two cases: If the information is correct, the details are saved on the database, if not the user has to retry with different details. If the process of saving the details is not executing correctly, the user must retry entaring the event details.

****

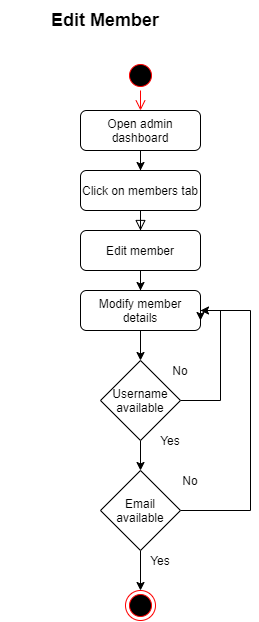
**4.Delete Member activity diagram**

Using our web applications admins have the option to delete members at any time by accessing the admin dashboard then choosing the user that needs to be deleted from the members tab,

****

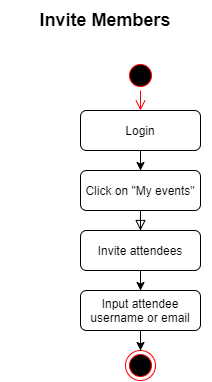
**5.Edit Member activity diagram**

Using our web application admins have the optin to edit the member details at any time by opening the admin dashboard then choosing the member who’s information needs to be changed. If after moddifying the details any of them are already in use, the admin will have to provide new credentials. After the procedure is successful the new details are saved on the database.

****

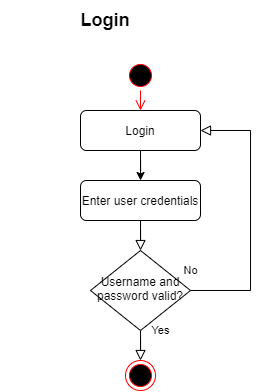
**6.Invite Member activity diagram**

Any user that is hosting an event has the option to invite members to that event by accessing the my event tab and choosing the specific person.

****

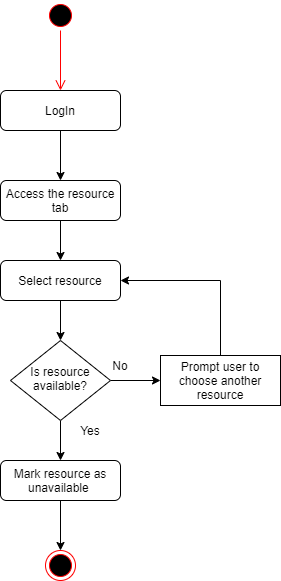
**7.Log In activity diagram**

If a user accesses our web application they will first have to log in into their account by accessing the log in button and introducting their credentials. If the credentials are inside of our database the user will be directed to the Home page, otherwise it will be asked to try the Log In procedure again.

****

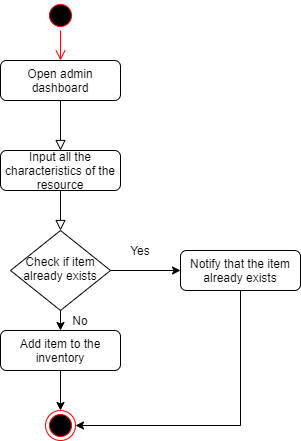
**8.Book Resource activity diagram**

Any user that is accessing our web application can book resources for their events by logging in and selecting the resource they wish to use. If the resource is not available then the use will be prompted to pick another. Otherwise the resource is booked and will become unavailable for other users.

****

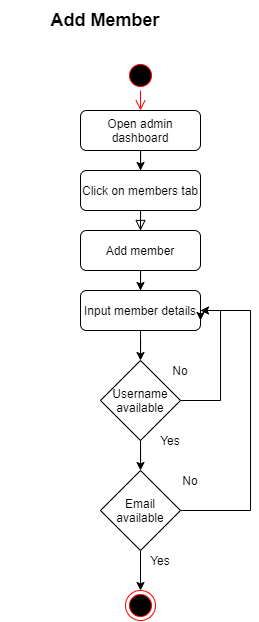
**9.Add Resource activity diagram**

Using our web application an admin can add new resources by accessing the admin dashboard and inserting the new resource into the inventory. If the item already exits the admin will be notified.

****

**10.Add Member activity diagram**

Using our web application an admin can add new members by firstly accessing the admin dashnboard, then selecting the members tab and the option “add member”. If the credentials chosen by the admin are already in use he will have to use new ones.After this step is executed correctly the new member will be added to the database.

****

## Design Rationale

Discuss the rationale for selecting the architecture described in 3.1 including critical issues and trade/offs that were considered. You may discuss other architectures that were considered, provided that you explain why you didn’t choose them.

We had chosen the MVC architectural pattern due to us being familiar with it.The most important characteristic is the separation of concerns:

-Model represents the data.

-View handles the User Interface.

-Controller handles the users requests.

# DATA DESIGN

## Data Description

Explain how the information domain of your system is transformed into data structures. Describe how the major data or system entities are stored, processed and organized. List any databases or data storage items.

Our web application uses SQL Server for data storage and Entity Framework for converting the data from the database into C# objects and vice versa. We followed Entity Framework's code first approach for our project so we can focus on creating classes for our domain entity rather than design our database first and then create the classes which match our database design.

## Data Dictionary

Alphabetically list the system entities or major data along with their types and descriptions. If you provided a functional description in Section 3.2, list all the functions and function parameters. If you provided an OO description, list the objects and its attributes, methods and method parameters.

--COMPLETE THIS--

# COMPONENT DESIGN

In this section, we take a closer look at what each component does in a more systematic way. If you gave a functional description in section 3.2, provide a summary of your algorithm for each function listed in 3.2 in procedural description language (PDL) or pseudocode. If you gave an OO description, summarize each object member function for all the objects listed in 3.2 in PDL or pseudocode. Describe any local data when necessary.

--COMPLETE THIS--

# HUMAN INTERFACE DESIGN

## Overview of User Interface

Describe the functionality of the system from the user’s perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback information that will be displayed for the user.

The web application will have a vast array of different functionalities such as:

-Users can create/login inside of their account

-Users have the capacity to create/edit events

-Users can select the members they wish to invite for their events

-Subscription payment system

-The ability to add/edit members if you are an administrator

-The ability to rent resources for certain events

-Viewing the participants in a certain event

-The option to add fees/donations

-The ability to see the reserved resources and events through a calendar

Our web application will be composed of the fallowing 5 web pages:

1.Home Menu

2.Account Menu

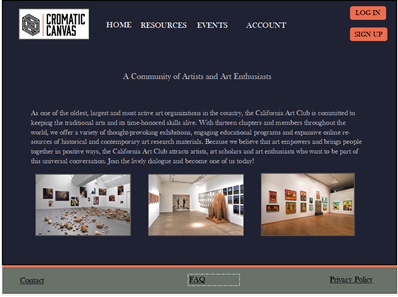
3.Event Menu

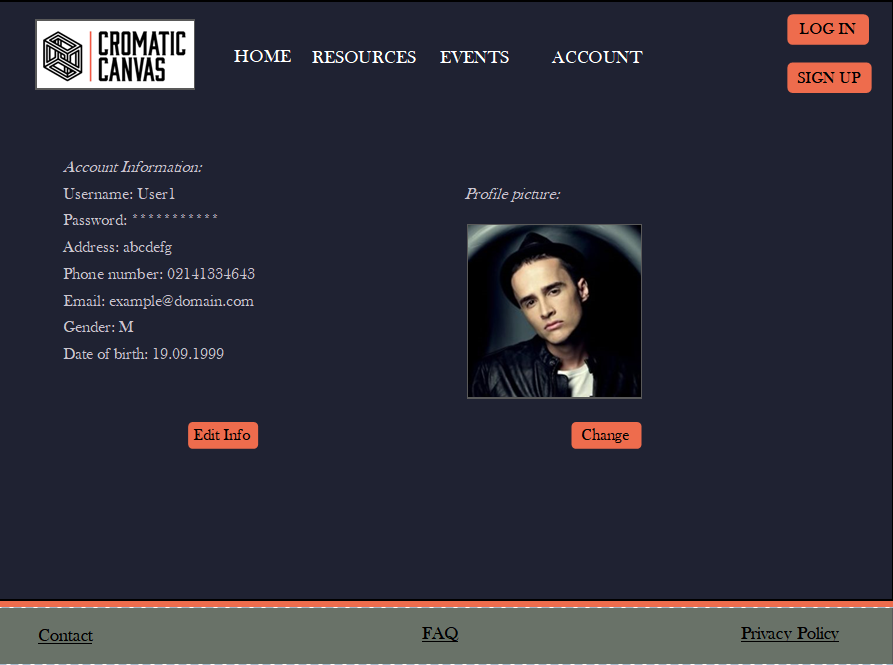
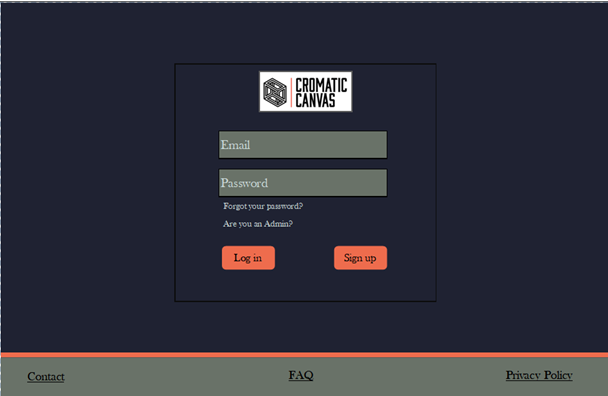
4.Log In

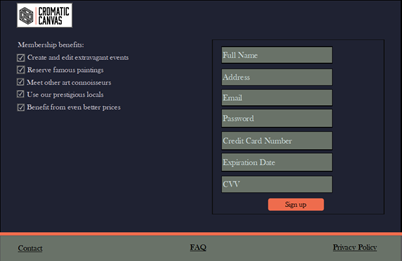
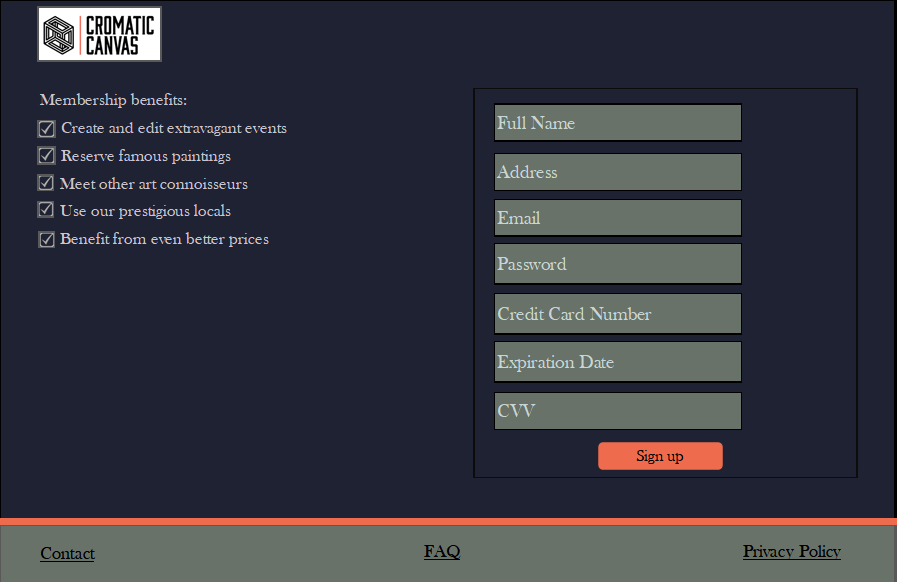
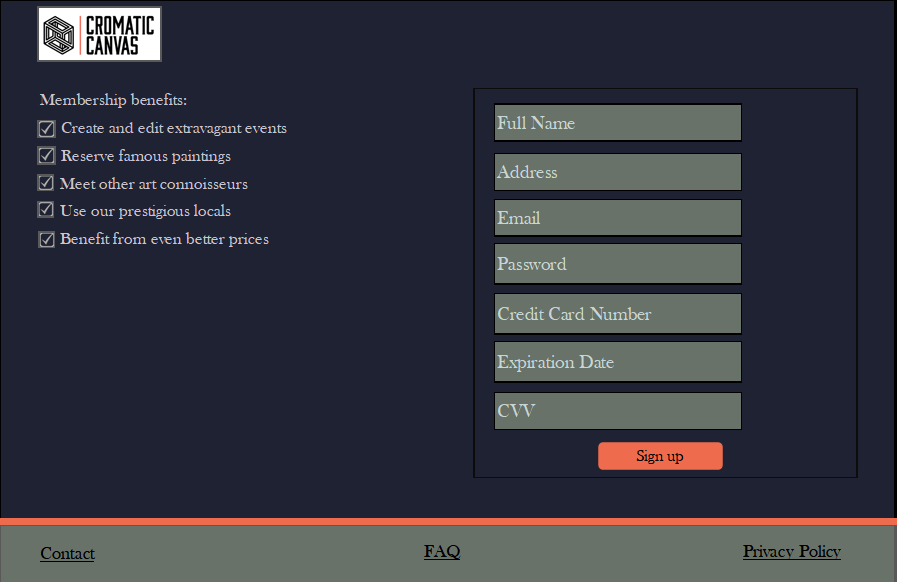
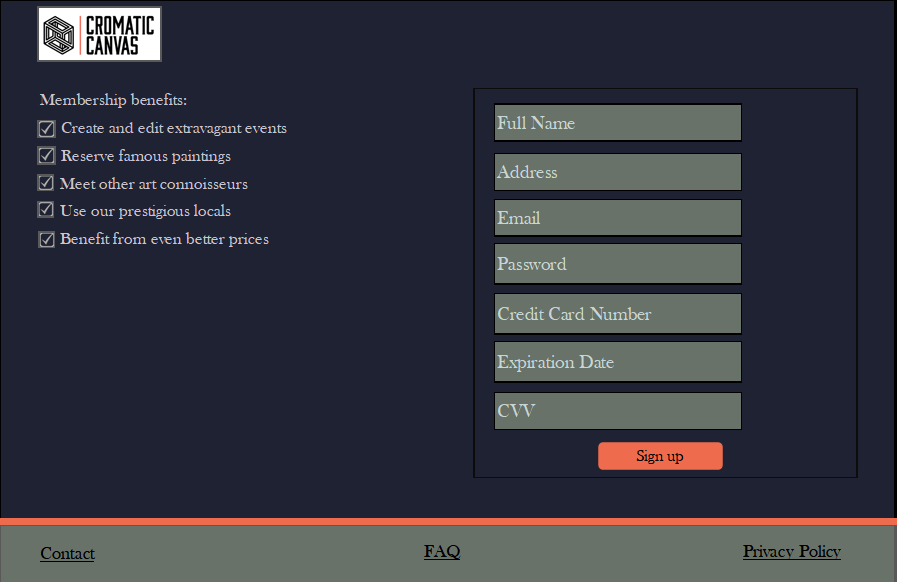
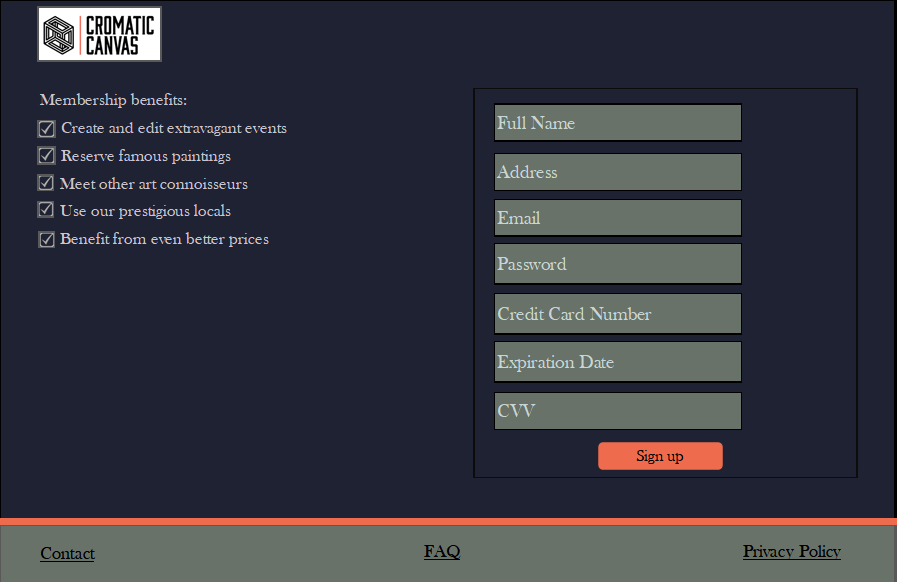
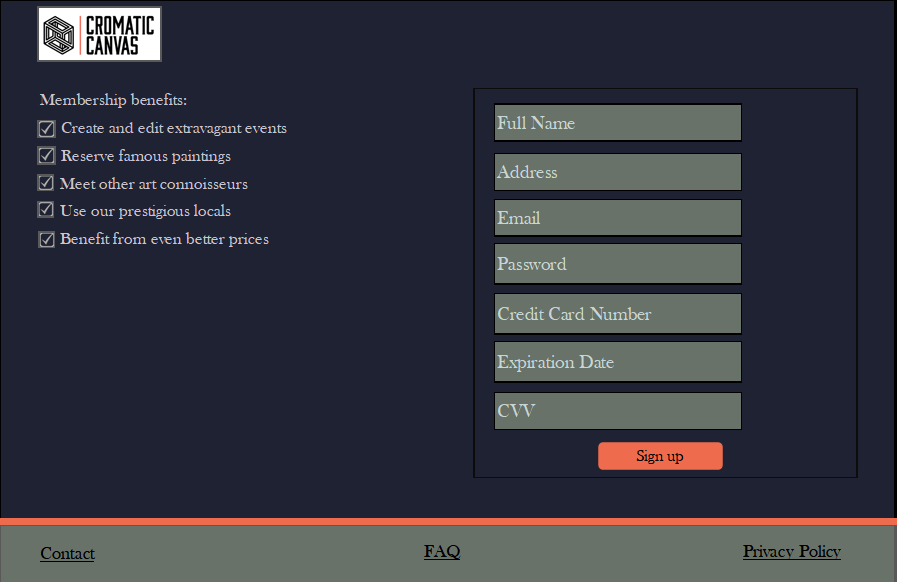
5.Sign Up

## Screen Images

Display screenshots showing the interface from the user’s perspective. These can be hand­ drawn or you can use an automated drawing tool. Just make them as accurate as possible. (Graph paper works well.)





## Screen Objects and Actions

A discussion of screen objects and actions associated with those objects.

--COMPLETE THIS--

# REQUIREMENTS MATRIX

Provide a cross­reference that traces components and data structures to the requirements in your SRS document.

Use a tabular format to show which system components satisfy each of the functional requirements from the SRS. Refer to the functional requirements by the numbers/codes that you gave them in the SRS.

--COMPLETE THIS--

# APPENDICES

1. http://www.cs.concordia.ca/~ormandj/comp354/2003/Project/ieee­SDD.pdf [↑](#footnote-ref-1)