COSC 625 Project Document

Fashion Advisor App

by  
Matthew Benson, Alison Jamison & Hector Caceres

Spring 2017

Frostburg State University

# Table of Contents

[Table of Contents 2](#_Toc483086041)

[1. Scope 3](#_Toc483086042)

[1.1. System Objectives 3](#_Toc483086043)

[1.2. System Requirements 3](#_Toc483086044)

[1.3. Design Constrains 3](#_Toc483086045)

[2. Data Design 3](#_Toc483086046)

[2.1. A Data objects and resultant data structures 3](#_Toc483086047)

[2.2. File and database structures 3](#_Toc483086048)

[2.2.1. External file structure 3](#_Toc483086049)

[3. Architectural Design 4](#_Toc483086050)

[3.1. Use Case Diagram 4](#_Toc483086051)

[3.2. Class Diagram 5](#_Toc483086052)

[3.3. Sequence Diagram 5](#_Toc483086053)

[3.4. State Diagram 6](#_Toc483086054)

[4. Interface Design 6](#_Toc483086055)

[4.1. Human-machine interface specification 6](#_Toc483086056)

[4.2. Human-machine interface design rules 6](#_Toc483086057)

[4.3. External interface design 7](#_Toc483086058)

[4.3.1. Interfaces to external data 7](#_Toc483086059)

[4.3.2. Interfaces to external systems or devices 7](#_Toc483086060)

[5. Procedural design 7](#_Toc483086061)

[5.1. Module descriptions 7](#_Toc483086062)

[5.1.1. Main Tab 7](#_Toc483086063)

[5.1.2. Plan Tab 7](#_Toc483086064)

[5.1.3. Wardrobe Tab 8](#_Toc483086065)

[5.1.4. Preferences Tab 8](#_Toc483086066)

# Scope

## System Objectives

An android app that helps the user to choose a perfect outfit that fulfill the requirements for a specific occasion. With this app, the user can take pictures of the clothes he owns and store them to facilitate the decision of what to use, from shoes to hats.

## System Requirements

The app will have the ability to:

1. Take and store pictures of single pieces of clothing with its specific information. Such as:
   1. Category
   2. Description
   3. Color
   4. Weather capabilities
   5. Activities to which it can be worn
2. Retrieve information about the weather, and based on it set a list for the best options for that weather.
3. Retrieve information about the activities that the user will do, and set a list for the best options for that occasion.
4. Ability to swipe between the options to allow the user to correct and set his own preferences.
5. Store user preferences and give it priority for next uses.

## Design Constrains

Due to this is the first version, we won’t be able to implement image recognition, and the app will rely on the user input to set the properties of each specific piece of clothing.

# Data Design

## A Data objects and resultant data structures

The principal data object in our design is called: “Articles”, which refers to every clothing item stored and used for the system.

This object has the same variables that are going to be stored in JSON.

We also have an object called “Closet”, in this object we store the “Articles”. We can add or remove “Articles” in any moment. We also use the “Closet” to get the outfit based on the temperature, the formality, and the user preferences.

## File and database structures

### External file structure

We used JSON files to store all the data, even the photos of the clothing items; however, as JSON only supports text, so we had to convert the binary file (image) into a string, and then we can add it to a JSON field.

#### Logical structure

Every article we are storing as JSON file has the following structure and properties:

"id"

"texture"

"idealTemp"

"formality"

"name”

"color"

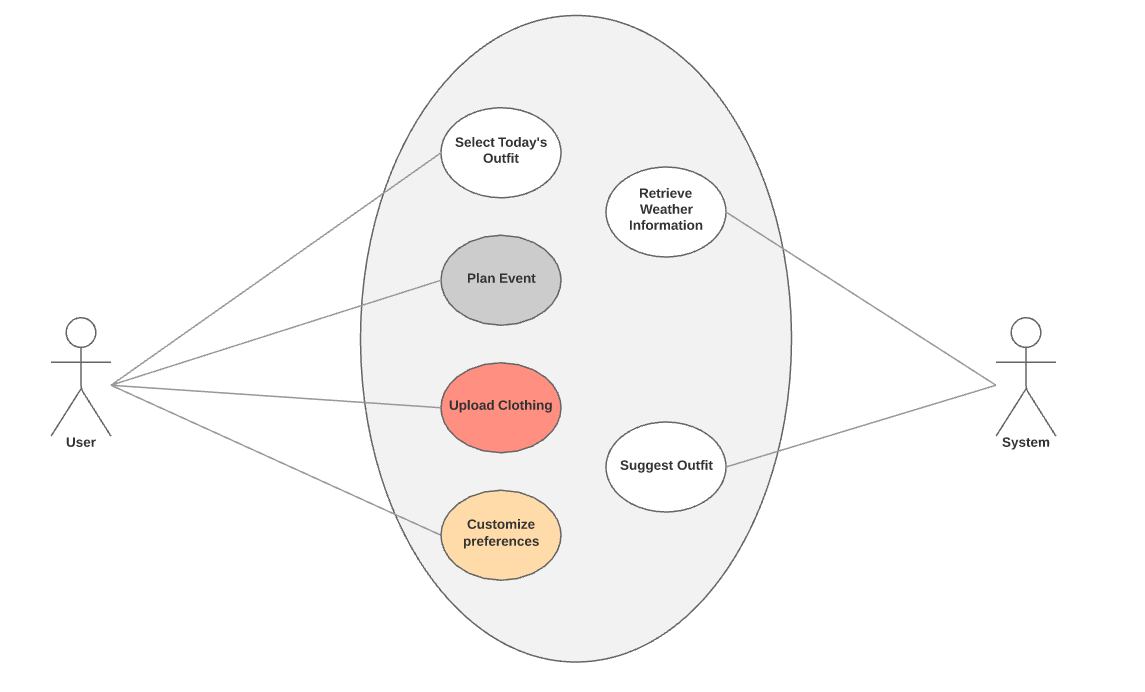
"patterned"

"img"

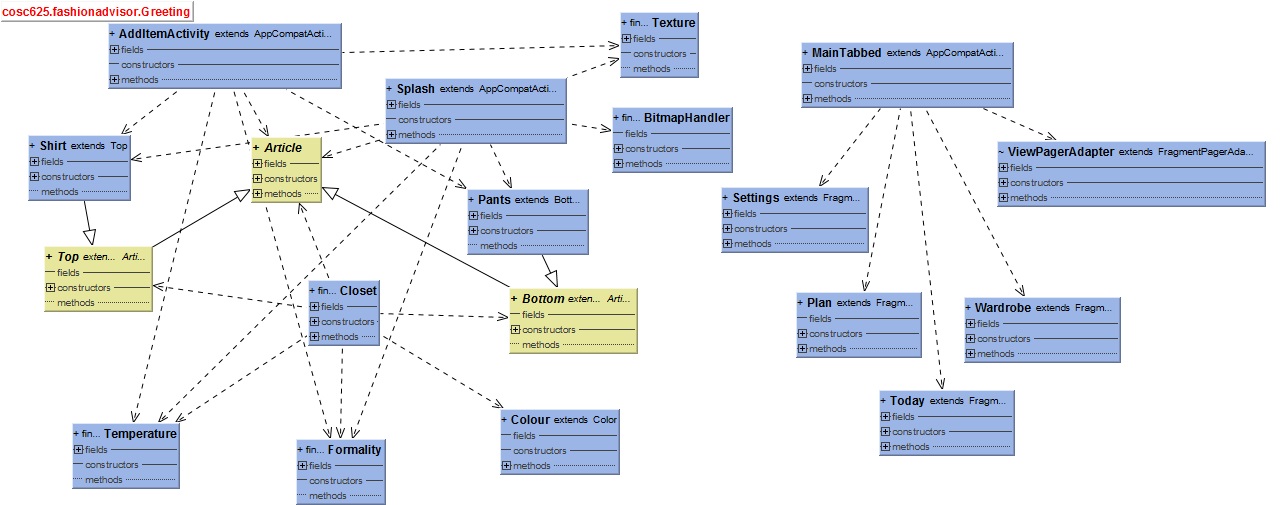
# Architectural Design

## Use Case Diagram

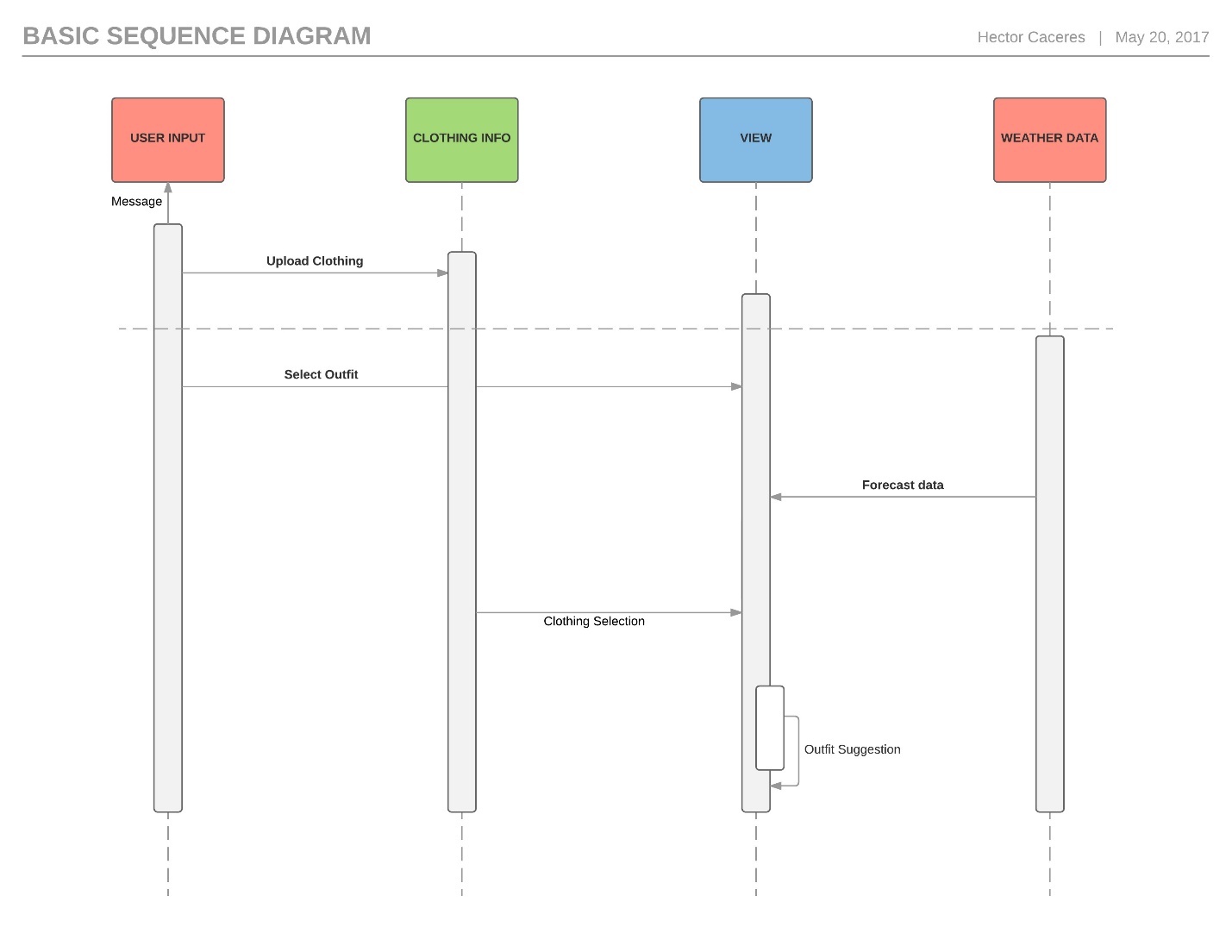
In our Use Case Diagram, we have two actors: user and system, and their interaction are simple:



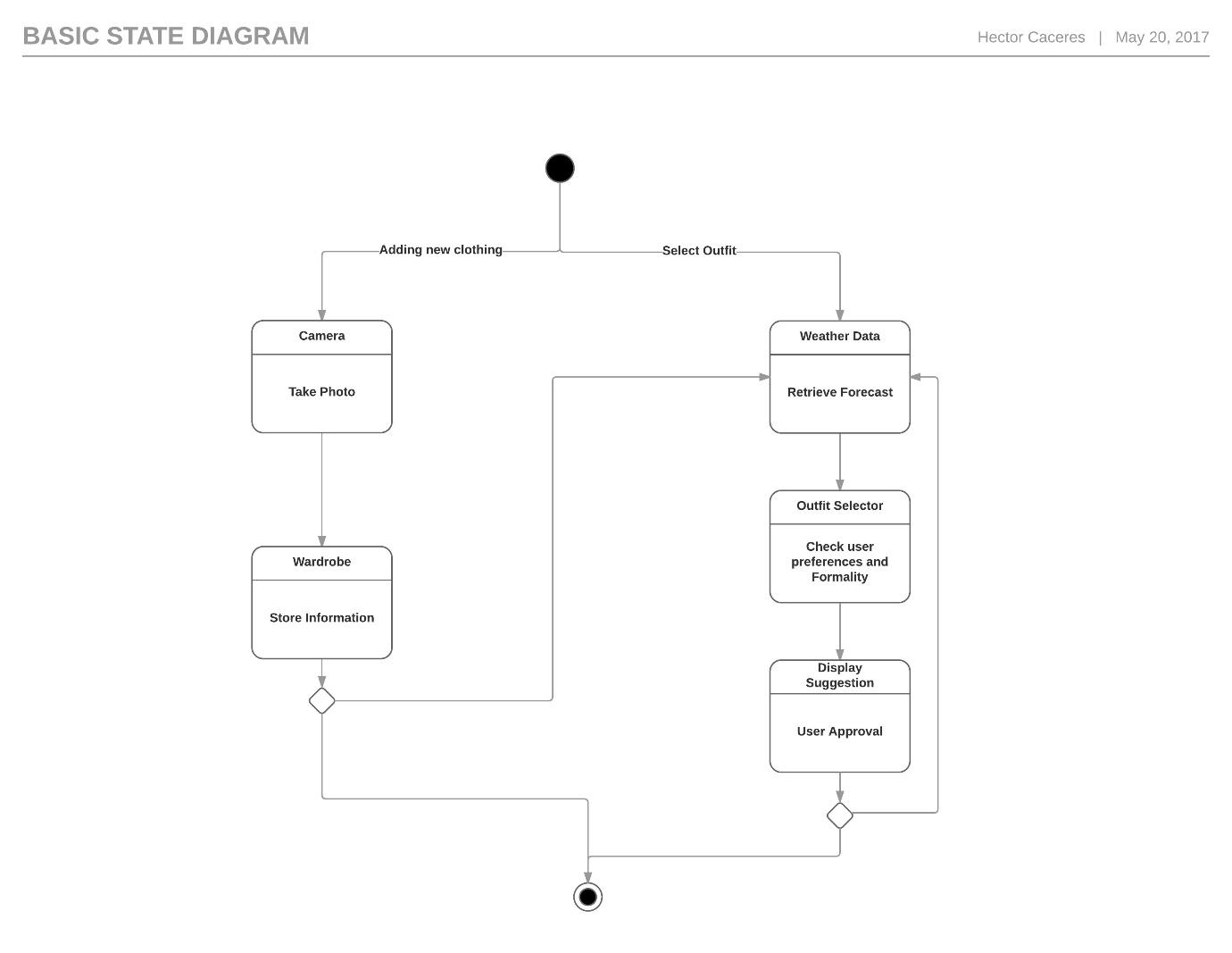
## Class Diagram



## Sequence Diagram



## State Diagram



# Interface Design

## Human-machine interface specification

The design of the interface is intended to be as similar to the experience of actually sorting through one’s wardrobe as possible while maintaining a futuristic, intuitive feel. The application is divided into four tabs: Today, Plan, Wardrobe, and Settings. The overarching color scheme will be primarily blue and white.

The Today tab is the outfit building screen. The outfit builder will be a series of carousel interfaces that the user may swipe back and forth through to select an item of clothing for each category. The Plan tab will have a calendar interface that allows the user to plan an outfit ahead of time. The Wardrobe tab will include a list of all articles in the user’s wardrobe, presented with an image and a short description of each item. Finally, the Settings tab will show a list of global settings pertinent to the user.

Throughout the application, whenever a user creates, modifies, or deletes an item of clothing, or modifies a user setting, a confirmation of success will be shown on the bottom of the screen.

## Human-machine interface design rules

The rules of the interface design will fit as closely as possible with the Material Design specification guidelines created by Google.

## External interface design

The user will be able to use the touch screen to select items of clothing, swipe through the carousel, and access the tabs of the application. If the user’s device provides a hardware keyboard, that can be used to enter text data; in the absence of a hardware keyboard, an on-screen keyboard will be displayed that the user can interact with via the touch screen.

### Interfaces to external data

The system will store data about the user settings on a locally hosted file called user.info. This file will be stored in the folder assigned to the application by the host system.

### Interfaces to external systems or devices

Information about the clothing contained in the user’s wardrobe will be stored in a remote database that is access via a username and password from the user.

# Procedural design

## Module descriptions

We created 4 different modules (Tabs) to manage the functionality of the app: Main, Plan, Wardrobe, and Preferences.

### Main (Today) Tab

**Abstract**: This is the home tab, where the user can select the formality level and ask to make an outfit for the day, the system will check the weather forecast and the formality level the user gave and select the clothing that meets those parameters.

**Implementation**:

**Exports**: N/A

**Imports**: N/A

**Pre Conditions**: In order to be able to suggest an outfit, items of clothing need to be stored in the wardrobe.

**Post Conditions**: N/A

**Input**: Weather Forecast, Formality Level, and User preferences.

**Output**: Outfit Suggestion.

### Plan Tab

**Abstract**: In this tab, the user will be able to plan an outfit ahead in time. By choosing a date in the future the system will check the weather forecast for that date and suggest an outfit. This is helpful in case the user need some preparation in a special occasion. This functionality works with the same logic as the today’s outfit, but using a future prediction of weather.

**Implementation**:

**Exports**: N/A

**Imports**: N/A

**Pre Conditions**: In order to be able to suggest an outfit, items of clothing need to be stored in the wardrobe. Also, the system needs to be able to retrieve a future forecast.

**Post Conditions**: N/A

**Input**: Future Weather Forecast, Formality Level, and User preferences.

**Output**: Outfit Suggestion.

### Wardrobe Tab

**Abstract**: In this tab, the user will be able to add new item to his own clothing collection. This functionality uses the phone camera to take a photo of the item and store it with some information, that the user needs to give as input:

* Item name.
* Type.
* Texture.
* Ideal temperature.
* Formality level.
* Color.

**Implementation**: To store the images as strings we used the function:

BitmapHandler.BitMapToString(image).

**Exports**: N/A

**Imports**: N/A

**Pre Conditions**: Phone camera functions.

**Post Conditions**: N/A

**Input**: Photo and description of the item.

**Output**: N/A

### Preferences Tab

**Abstract**: In this tab, the user will be able personalize the app to his own preference. The user enters his/her name, gender, and can setup the ideal temperature, in order to be used as input when choosing the outfit.

**Implementation**:

**Exports**: N/A

**Imports**: N/A

**Pre Conditions**: N/A

**Post Conditions**: N/A

**Input**: User input.

**Output**: N/A