# What we did

Using Angular JS, the wardrobe assistant was created as a web application using J2EE framework for services and connecting to the database. From our interpretation of the requirements documentation provided, our focus has been on implementation of the wardrobe system which allows for the user to add, edit, view, and remove the clothing items that they own.

# System Functionalities

## Functionalities covered in implementation

The functionalities implemented for the wardrobe assistant, with the focus on high-priority features, are as follows. A simple and easy-to-navigate user interface houses all of the functionality created for the wardrobe assistant. The user can log in from the home page, and once logged in, can access their account settings or log out no matter what screen they are accessing. An easy-to-access menu presents all of the options available in the system so that everything is transparent to the user.

The main goal of our implementation, due to the fact that all other requirements (such as the clothing suggestions or outfit builder) stem from it is the wardrobe management system. We have implemented the ability to add, remove, and edit their wardrobe which is automatically sorted by the type of clothing item. To support the suggestion/recommendation features as well as the measurement feature, information such as size, style, color, and pattern were included.

Since wardrobes are unique to each person, another high priority feature was the ability to have user accounts. Users have the ability to register for an account, log in to the system, log out from the system, and edit their account information.

## Important changes from original specification.

Although the specification lists providing measurements as a part of the recommendation/suggestion system, (3.5.1.8) it was instead implemented as a part of the wardrobe item itself. Due to time constraints the recommendation/suggestion system was not fully implemented, but the importance of storing measurements was still considered. So, in order to still satisfy the use case, it was moved to the item editor instead.

Removing items also was implemented differently. Instead of having multiple steps to remove an item (selecting the item, selecting remove, and confirming the removal), a remove mode allows for the user to quickly remove items in succession. This was done to reduce the number of screens the user needs to view (and would be more mobile/user friendly) in order to accomplish the task of removing an item from the wardrobe.

Since item suggestions have not been implemented, the outfit builder is not based on suggested items. The outfit builder allows for the user to group clothing items from their inventory instead of a set of example items. The user can visualize potential outfit combinations with their own clothes instead of clothes they would have to go out and try to buy.

## Requirements Excluded from original specification

The recommendation of wardrobe items and dynamic suggestions were excluded from this implementation. Since recommendations are unique to each user, there were no specific guidelines in the requirements document, and we do not know the target culture of users, it was too difficult to create a dynamic suggestion system as specified. However, each wardrobe item has characteristics that allow for a future system to be developed. Information such as color, pattern, formality (casual/semi-formal/formal), and the type of clothing item were included.

# 

**Implemented Functionality**

Login

Registration

SearchWardrobe by Category (Ex: Hats, Tops by clicking the tab set )

Add Item from the Main Landing page. Did not integrate the Add Item after invoking search or other functionality.

# How to build and run

Installation requires lots of software and some setup. If you do not want to do this, I can demonstrate this from my laptop.

Extract **Part2-Project.zip** to your file system. You will see wrs-client, wrs-server archives, readme file and sql-script file after extracting.

**Server Installation:**

Install postgres database and run at 5432 port. Use postgres/postgres as the userid and password for the database.

Create WRS database using pgAdmin tool.

Run the sql-script to create database objects.

Install Spring STS from <https://spring.io/tools/sts/all>

Extract wrs-server.zip to local file system.

Launch the STS and import the server project from local file system.

Run the server code from Boot Dashboard.

**Client Installation:**

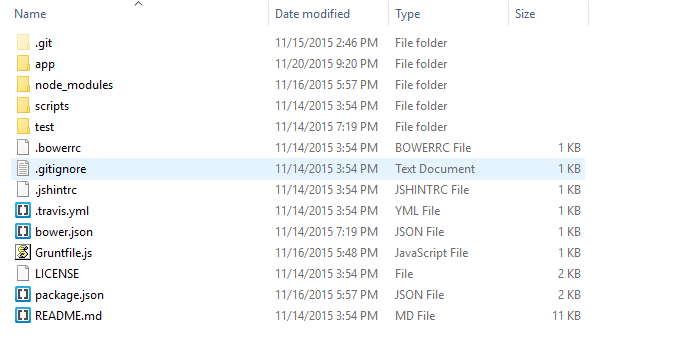
Install git from http://git-scm.com/download

Instal Node.js from <https://nodejs.org/en/>

Make sure bin directories of both software in the path

extract wrs-client.zip

open node.js command prompt and go to the root directory of the wrs-client as shown below execute following commands



Now run the following command

**npm install**

Now run the following command

**npm install grunt-cli**

To start the client application just do following

grunt

See the application from this url in Internet Explorer.

http://localhost:8000/app