

### Robe Rental

- 3: Robe Rental
- Prathyusha Butti, Priyamvadha Gopalakrishnan, Ripan Chowdhury and Royan Tuscano

#### 1. Introduction

- We have developed a minimal viable system implementing the following features:
  - A GET API for the home page with the list of garments and images.
  - A “Personal Details” Tab under My Account Page for the user to enter the details.
  - A “Sell or Rent Your Garment” Tab under My Account Page for the signed in user to upload and enter the details of the garment he/she wants to sell or rent.
  - A GET API for “Personal Details” Tab to view the previously entered user details
  - A POST API for Image upload along with the Garment details.
- The user stories we have implemented for this iteration are as follows:
  - User Story 2 : As a seller, I want to rent my high-end garments so that I can continuously generate revenue.
  - User Story 5 : As a seller, I want to display the size and material of the garment so that the buyer can choose a garment based on the size and material.
  - User Story 6 : As a buyer/seller, I want to create a profile in the website so that I do not have to re-enter my information for payment every time I make a payment.
  - User Story 10 : As a seller, I want to upload a picture of the garment along with all the specifications so that the buyer is able to find all the information in one place.
- Overview of changes to problem definition, design, technology, team roles:
  - “No Changes”

#### 2. Customer Value and Problem Definition

- Overview of Customer Value: “No changes”

The primary customers fall under two types:

- Buyers who need a garment for one time use.
- Sellers who will rent their clothes.

## Unmet Customer Needs

- Customer needs:
  - A user/buyer wants to rent a garment for a low cost or for one-time use without paying full price for it.
  - A seller wants to rent or sell dresses they no longer want to use and make a business out of it.
- Desired overall experience:
  - To provide various categories for the buyer to select a garment of their choice.
  - To build a flexible business model that will have fair pricing depending on the number of days of rent for both the buyer and seller.
  - To provide profiles for both the buyer and seller to do the rental activity.
  - To integrate a payment system for purchases.
  - To address following type of internet customers
    - Product focused shoppers: Search bar or navigation that quickly shows the product customer is interested in buying.
    - Browsers and Researchers: Show customized products based on users' browsing history and past searches.
    - Bargain hunters: Give exclusive deals on products to this type of customers.
    - One time shopper: Offer incentive to visit again by offering rewards based on history.
- Unmet needs and user stories based on the unmet needs:
  - There are people who do not want to buy a dress for just one time use as they cannot afford to buy it, for example a wedding gown or prom dress. In such cases the person can rent the dress for a week for fraction of price of original dress .This will save the purchase cost for the buyer.
  - There are people who have too many designer clothes that they don't use, they can rent those clothes and generate revenue.
- Overview of Problem Definition : **“No changes”**
  - Two major customer opportunities we found and plan to address are:
    - Excessive items in a wardrobe.
    - Budget constraints holding back garment purchase.
  - Options for addressing these needs:
    - Website for connecting a potential garment buyer with a seller.
    - Mobile app to connect a buyer to a seller.

We chose to build a website first and then at a later point, make the website adaptive to screen size.
- Any changes from the previous report? **“No changes”**

### 3. Technology

#### System at the end of this iteration

- Goals for this iteration :
  - Working frontend and backend using MVC (Model View Controller) architecture.
  - Implementing the home page display of garments functions that bind the backend with frontend.
  - Implementing the ‘personal details’ tab where the user entered details are stored in the database.
  - Implementing the image upload functionality where the user can upload the image of their garment and see the preview of it.
  - Implementing the ‘sell or rent your garment’ tab where the user entered garment details along with the image is stored in database.
- “Personal Details” Tab under **My Account** Page for the website :

The screenshot shows the 'Personal Details' form within the 'My Account' section of the 'Robe Rental' website. The form is titled 'Personal Details' and 'Sell or Rent Your Garment'. It contains several input fields for user information:

- First Name \*: Linda
- Last Name \*: Bahr
- Email \*: linda@gmail.com
- Address Line 1 \*: 1234 E University Blvd
- Address Line 2: Apt 1234
- City \*: Tucson
- State \*: Arizona
- Zip Code \*: 85781

Below the form is a 'Create Account' button. To the right of the form, there is a link that says 'Image removed - Protected View - View'.

- “Sell or Rent Your Garment” Tab under My Account Page for the website :

Robe Rental
Home
How it works
Team
Cart
Sign In
My Account

Personal Details
Sell or Rent Your Garment

Title For Your Garment \*  
Sastaras Blue Gown

Rent Price \*  
15


Buy Price \*  
30

Size of the garment \*  
M

Accepts only in the format S,M,L,XL,XXL,XXXL  
Material Description \*  
Women Navy Blue & Pink Printed Fit and Flare Dress

Type of the Material \*  
Cotton

Garment Care \*  
Hand-wash

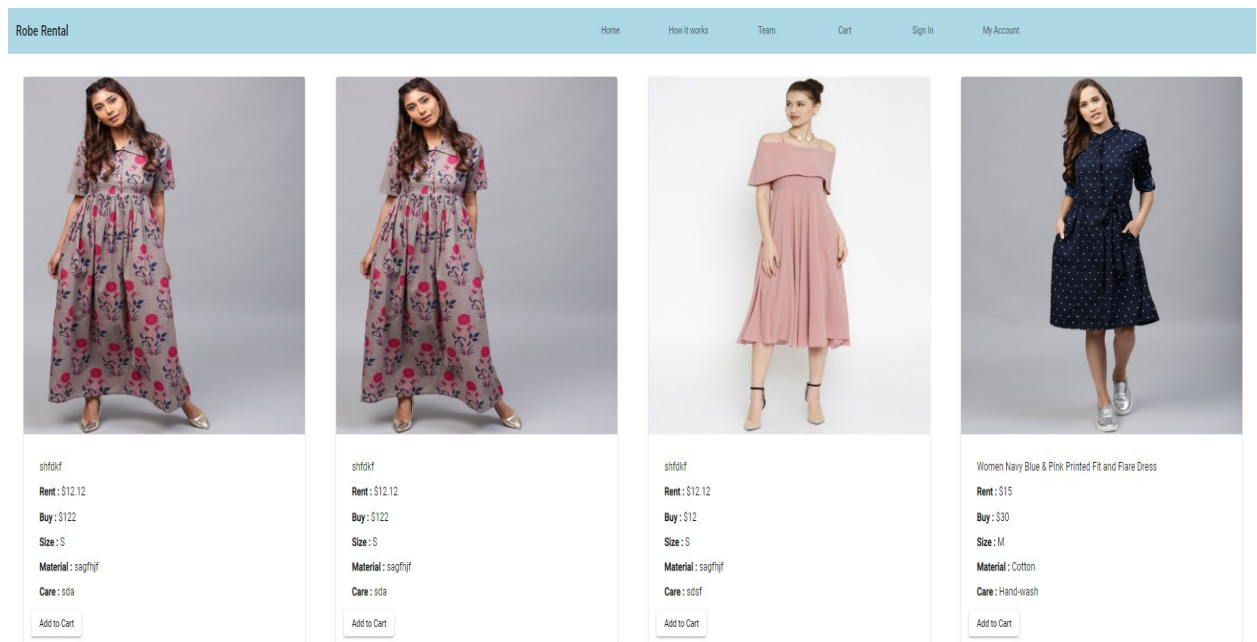


Select Image  
Choose File 00fa04f-c0b...#4080-1.jpg

ProjectStatus2 - Protected View - Word

Sell or Rent your Garment

- Home Page for the website with GET API integration:



- **Tests we have run are as follows:**

- We have run basic unit test cases with each module implemented which can be seen as the spec.ts file for each component and service file.
- We have run regression tests on each existing module after adding new features.
- We have run manual tests on the application and logged the results in google sheets as follows:

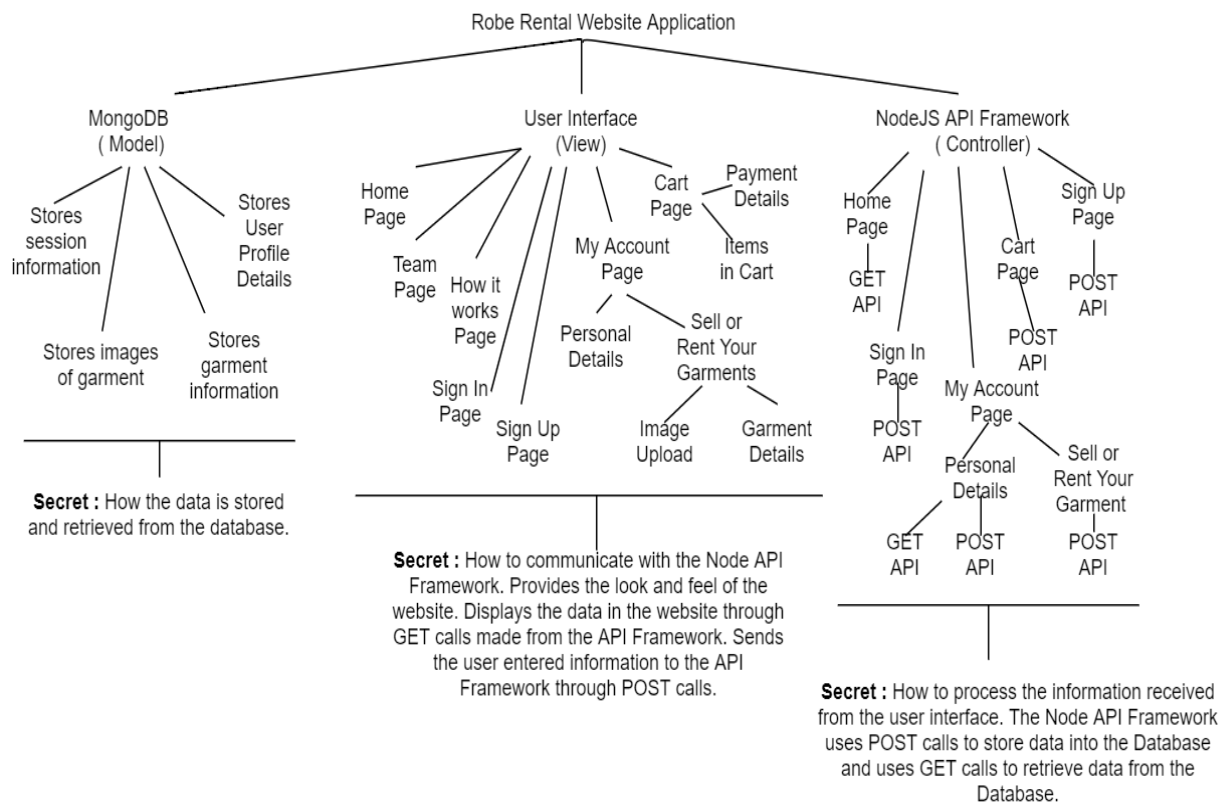
[https://docs.google.com/spreadsheets/d/1WHFJXMZ\\_ISyXC3QH-vWQMHL-1JhGgMryJ12aWaAxIz0/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1WHFJXMZ_ISyXC3QH-vWQMHL-1JhGgMryJ12aWaAxIz0/edit?usp=sharing)

- We have covered the following scenarios in test cases for this iteration :
  - User is able to see the list of garments in Home page
  - User is able to enter his personal details
  - User is able to upload the image and see a preview of it
  - User is able to enter the garment related information

- How many lines of code have you written as a team?

- The database server has around 400 lines of code in total.
- The front-end has around 1800 lines of code in total.

## Module Hierarchy



## Module Guide

### 1. MongoDB (Model)

The MongoDB module acts as the “Model” in our Model View Controller Architecture. This module includes programs that store session information, store images of garment, store garment information and store user profile information. Its secret of this module is how data is read and written to the cloud database. Programs in the MongoDB module are used by programs in the Node API Framework to read data and write data and then it is sent to the User Interface to display the data.

#### 1.1 Store Session Information

##### -Textual Description

- The responsibility of this module is to store all the session related information in the database.
- By hiding how it stores session information in the database, we made it easier to incorporate any future changes to the logic behind the session storage.

##### -Service Provided

- A ‘Store Session’ service is provided to the other modules through this module.

##### -Error and Exception Handling

- Network Disconnect
- Session Information expired
- Missing Session Data

##### Rationale:

Secrets: How data is stored in the database.

#### 1.2 Store Garment Images

##### -Textual Description

- The responsibility of this module is to store uploaded images of garment in the database.
- By hiding how it stores images in the database, we made it easier to incorporate any future changes to the logic behind the images storage.

##### -Service Provided

- A ‘Store Garment Image’ service is provided to the other modules through this module interface

##### -Error and Exception Handling

- Network Disconnect
- Unidentifiable image format
- Image size too large

**Rationale:**

Secrets: How image is stored in the database.

**1.3 Store Garment Information****-Textual Description**

- The responsibility of this module is to store garment information entered by the seller in the database.
- By hiding how it stores garment information in the database, we made it easier to incorporate any future changes to the logic behind the garment details storage.

**-Service Provided**

- A 'Store Garment Information' service is provided to the other modules through this module interface

**-Error and Exception Handling**

- Network Disconnect
- Unidentifiable information text
- Missing required information

**Rationale:**

Secrets: How garment information is stored in the database.

**1.4 Store User Profile Information****-Textual Description**

- The responsibility of this module is to store user profile information in the database.
- By hiding how it stores profile information in the database, we made it easier to incorporate any future changes to the logic behind the user profile information storage.

**-Service Provided**

- A 'User Profile Store' service is provided to the other modules through this module interface

**-Error and Exception Handling**

- Network Disconnect
- Missing required information
- Duplicate user profile

**Rationale:**

Secrets: How user profile is stored in the database.

**Note:**

We chose MongoDB as the Data Model as it provides fast performance, seamless data migration and flexible deployment.

- ❑ Owner of the entire database module: Royan Tuscano

## 2. User Interface (View)

The User Interface module acts as the “View” in our Model View Controller Architecture. It handles communication with the Node API Framework and provides a look and feel for the Robe Rental website using HTML, CSS, JavaScript and Angular Framework. It displays the data in the website through GET calls made from the API Framework. It sends the user information entered through the API Framework through POST calls.

This module includes all the programs that displays garments available for rent (through the ‘Home’ page, Frequently Asked Questions (through the ‘How it Works’ page), Team information, Cart information, Sign In and a ‘My Account’ page that provides the option to enter Personal Details and Sell or Rent out garments.

Its secrets include how each page interacts with the users, how it handles GET calls and POST calls using Node JS API Framework Module and also hides the logic behind these pages using Angular Framework.

### 2.1 Home Page

#### -Textual Description

- The responsibility of this module is to display all the garments that are available for rent or purchase.
- By hiding the logic behind how it reads from the database and displays garment images and details, we made it easier to change the logic behind how this page works in the future.

#### 2.1.1 Add To Cart

#### -Service Provided

- All Add To Cart services and garment list services are provided to the user by this module, under the Home page

#### -Error and Exception Handling

- Network Disconnect
- Garment Unavailable
- Cart Unavailable

#### Rationale:

Secrets: How available garments are displayed to the user  
How ‘Add To Cart’ service works

### 2.2 How It Works Page

#### -Textual Description

- The responsibility of this module is to display Frequently Asked Questions and their answers to help the users with their questions.
- By hiding how it reads FAQs from the database, we made it easier to change the logic behind how this page works in the future.



**-Service Provided**

- A 'How It Works' page is displayed to the user through this module interface.

**-Error and Exception Handling**

- Page Unavailable

**Rationale:**

Secrets: How information is read from the database and stored in the database.

**2.3 Team Page****-Textual Description**

- The responsibility of this module is to display our Team's background and experience to the users.
- By hiding how it reads Team information from the database, we made it easier to change the logic behind how this page works in the future.

**-Service Provided**

- A 'Team' page with details about each team member of this project is displayed to the user through this module interface.

**-Error and Exception Handling**

- Page Unavailable

**Rationale:**

Secrets: How Team data is read from the database.

**2.4 Cart Page****-Textual Description**

- The responsibility of this module is to maintain an updated list of garments chosen by the buyer for rent or purchase, by using the 'Add to Cart' button available in the Home page module. This information is displayed in the 'Items in Cart' page.
- This module is also responsible for handling Payment services when the items in a cart are checked out. This page is currently non-functional.
- By hiding how it reads Cart information from the database, we made it easier to change the logic behind how this page works in the future.

**2.4.1 Items in Cart****-Service Provided**

- All garments added to the cart by the buyer are displayed to the user through this service and it interacts with the 'Add to Cart' button available in Home Page module.

**-Error and Exception Handling**

- Network Unavailable
- Items in Cart page Unavailable

**Rationale:**

Secrets: How Cart data is read from the database.

**2.4.2 Payment Services****-Service Provided**

- No services are currently provided by the Payment module at this point.

**2.5 Sign In Page****-Textual Description**

- The responsibility of this module is to allow users to Sign In to the Robe Rental Service.
- It allows registered users to access their Account Information which is available in the 'My Account' module.
- The Sign In module accepts user input for Email and Password.
- By hiding how it validates user login credentials with the database on the Sign In module, we made it easier to change the logic behind how this page works in the future.

**-Service Provided**

- The Sign In service interacts with the 'MongoDB' module by calling the 'Store User Profile Information' service.

**-Error and Exception Handling**

- Page Unavailable
- Missing user data input
- Authentication Failed
- Duplicate User Profile

**Rationale:**

Secrets: How user data is read from the database.

**2.6 Sign Up Page****-Textual Description**

- The responsibility of this module is to allow new users to create a profile, so that they can start using the services provided by the Robe Rental application.
- By hiding how it stores user profile information in the database on the Sign Up module, we made it easier to change the logic behind how this page works in the future.

**-Service Provided**

- A Sign Up service is provided to the user through this sub module by allowing new users to create a new profile.
- This module accepts First Name, Last Name, Email, Password information from the user.
- The Sign Up service interacts with the NodeJS API Framework module for data storage

**-Error and Exception Handling**

- Page Unavailable
- Missing user data input
- Authentication Failed
- Duplicate User Profile

**Rationale:**

Secrets: How user password is encrypted before storing in the database

How new user profile information is stored in the database.

**2.6 My Account Page****-Textual Description**

- The responsibility of this module is to allow registered users to access their Account Information at any time. It allows the users to view and edit their 'Personal Details' and also 'Sell or Rent their Garment'.
- By hiding how it interacts with the database to store and retrieve Personal Details, we made it easier to change the logic behind how this logic works in the future.
- By hiding how it interacts with the database to store and retrieve Garment images and garment information, we made it easier to change the logic behind how this logic works in the future.

**2.6.1 Personal Details****-Service Provided**

- All registered users are provided with a Personal details service where all registered users can view and edit personal details.
- This module accepts First Name, Last Name, Email, Address Line 1 and 2, City, State, Zip code information from the user.
- This module interacts with the NodeJS API Framework module to store and retrieve from the database.

**-Error and Exception Handling**

- Page Unavailable
- Unable to find user
- Invalid user input

**2.6.2 Rent or Sell Your Garment****-Textual Description**

- The responsibility of this module is to provide registered users with Image Upload service and add Garment Details service.

**2.6.2.1 Image Upload****-Service Provided**

- All registered users are provided with a 'Image Upload' service in which registered users can upload garment images to rent or sell.
- This module accepts Garment Image uploads from the user.

- This module interacts with the NodeJS API Framework module to store and retrieve images from the database.

#### **-Error and Exception Handling**

- Page Unavailable
- Unidentified image format
- Image too large

#### **Rationale:**

Secrets: How Garment Images are stored and retrieved from the database.

### **2.6.2.2 Garment Details**

#### **-Service Provided**

- All registered users are provided with a 'Rent or Sell Garment' service in which registered users can add garment details to rent or sell.
- This module accepts Title For Your Garment, Rent Price, Buy Price, Size of the garment, Material Description, Type of Material, Garment Care information from the user.
- This module interacts with the MongoDB module (Model) to read and write to the database.

#### **-Error and Exception Handling**

- Page Unavailable
- Invalid Input
- Incomplete Data

#### **Rationale:**

Secrets: How Garment Details is stored and retrieved from the database.

- ❑ Owner of the entire User Interface module: Prathyusha Butti

## **3. NodeJS API Framework (Controller)**

#### **-Textual Description**

- The responsibility of this module is to handle all the GET calls and POST calls made from the User Interface Module.
- The NodeJS API Framework acts as the 'Controller' in our Model View Controller architecture. This module determines how to process all the information received from the user interface.
- The Node API Framework uses POST calls to store data into the database. It uses the Get calls to retrieve data from the database.
- This module is shared by the services in the User Interface module, viz, Home Page, Sign In Page, My Account Page, Personal Details Tab, Sell or Rent Your Garment Tab, Cart Page and the Sign Up page.
- By hiding how this module processes user information and how it interacts with the User Interface Module, we made it flexible to change these logics in the future.

**-Service Provided**

The services provided by this module include handling all the GET API and POST API calls from the User Interface Module.

- It processes the GET API calls originating from the Home Page
- It processes the POST API calls originating from the Sign In Page
- It processes the GET API calls and POST API calls originating from the 'Personal Details' tab available under the My Account Page
- It processes the POST API calls originating from the 'Sell or Rent Your Garment' tab available under the My Account Page
- It processes the POST API calls originating from the Cart Page
- It processes the POST API calls originating from the Sign Up page.

**-Error and Exception Handling**

- GET API Service Unavailable
- POST API Service Unavailable
- Database connection lost.

**Rationale:**

Secrets: How all the user information received from the User Interface module are processed.

How it uses GET and POST calls to store and retrieve information in the database.

- ❑ Owner of the entire Node JS API Framework module: Royan Tuscano

**4. Team**

- *Team member roles and contributions for this iteration :*
  - **Prathyusha** - As a Product Owner, she has prioritized which user stories needs to be implemented so that a minimum viable system is up and running. As Front-end developer she has worked with the image upload and preview. She has developed the Personal Details and Sell or Rent Your Garments Tab under My Account Page. She even developed the Home Page GET API calls with Back-end developer. She even contributed towards the project documentation by revising the module hierarchy and adding project related details. She maintains the Service Contract API document. She has contributed around 40% for this iteration.
  - **Priyamvadh**a - As a Scrum Master, she has conducted daily scrums. As a Manual Tester, she has written and executed the manual test cases for the second iteration. She even performed regression testing for the previous modules. She has completed the module hierarchy and module guide for this project. She has contributed around 20% for this iteration.
  - **Ripan** - As a Back-end developer he needs to develop the GET API to display the list of garments in the home page. But he could not succeed in doing the task. His contribution for this milestone is 0%.
  - **Royan** - As a Back-end developer he took the initiative of completing the previous iteration backlog item i.e GET API call to display the list of garments in home page. He even developed the GET and POST API for Personal Details Tab and POST API for Garment Information. He has contributed around 40% for this iteration.

- *Team member contribution by module hierarchy :*
  - MongoDB (Model) :
    - Store images of the garment to the database - Royan
    - Store the user profile details to the database - Royan
    - Store the session information to the database - Royan
    - Store the garment information to the database - Royan
    - Store the order details - Royan (Next Iteration)
  - User Interface (View) :
    - Home Page - Prathyusha
    - Team Page - Prathyusha
    - How it works Page
      - Content for the page is provided by Priyamvadha
      - HTML, CSS, Angular, JavaScript implementation done by Prathyusha
    - Sign In Page - Prathyusha
    - Sign Up Page - Prathyusha
    - Personal Details Tab under My Account Page - Prathyusha
    - Sell or Rent Your Garment Page under My Account Page - Prathyusha
    - Cart Page - Prathyusha (Next Iteration)
  - Node API Framework (Controller) :
    - Home Page - GET API - Royan
    - Sign In/Sign Up - POST API - Royan
    - My Account - GET and POST API - Royan
    - Cart - POST API - Royan (Next Iteration)

## 5. Project Management

- Plan for the next iteration and the rest of the semester
  - Payment module
  - Sort the Garments
- Track changes to design, test cases and code :
  - We are using Google Docs to maintain the API contract.
    - The link to the API contract is as follows:  
[https://docs.google.com/document/d/1hRxrKzGn-Iy87juDzL9V\\_BBJbUYksuX6qmn9i8vHpVQ/edit?usp=sharing](https://docs.google.com/document/d/1hRxrKzGn-Iy87juDzL9V_BBJbUYksuX6qmn9i8vHpVQ/edit?usp=sharing)
  - We are using Google Sheets to maintain the test cases and its results.
    - The link to the test cases sheets is as follows:  
[https://docs.google.com/spreadsheets/d/1WHFJXMZ\\_ISyXC3QH-vWQMHL-1JhGgMryJ12aWaAxIz0/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1WHFJXMZ_ISyXC3QH-vWQMHL-1JhGgMryJ12aWaAxIz0/edit?usp=sharing)
  - We are using GitHub to track the changes to the code
    - The link to the GitHub for both front-end and back-end code is as follows:  
<https://github.com/buttiprathyu/roberental>

## 6. Reflection

- What went well?
  - We were able to implement GET API for the list of garments for the home page which was carried over to this iteration from the previous iteration.
  - We were able to store and retrieve the user personal details.
  - We were able to upload and retrieve the garment information along with its specifications.
- What didn't go well?
  - We were not able to implement the sorting of the garments based on Size and Price as planned initially.
- For the features that were planned but not implemented, what were the issues?
  - Lack of availability of the assigned developer to do the sort functionality. Two developers were fully focused on the image upload functionality as it was very challenging to implement on both front-end and back-end side which consumed most of the time for this iteration.
- How do you plan to overcome the issues you encountered?
  - We will plan better and have regular status checks with daily scrum calls.
  - We have to make sure to attend all scrum calls and update the work done through email if there is a meeting conflict.
  - We plan to finish assigned tasks on time or ask for help to complete them.
  - We plan to emphasize more on obtaining user and team feedback after each module development with a short demo.
- Redefining the problem - **“No changes”**
- What will you do differently for the next iteration?
  - We will improve our sprint planning.
  - We will make it a practice to check in the code regularly to coordinate the deployment of the builds.
  - We will communicate to the Scrum Master of any impediments we have in the scrum calls.
  - We will take feedback a earlier then we are doing now.