**CS673 Software Engineering**

**Team 4 - Project Name**

**Software Design Document**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role(s) | Signature | Date |
| Duan Lin | Team leader / QA leader | *Duan Lin* | 09/24/2020 |
| Hussain Alibrahim | Configuration leader / Front-End Designer 1 | *Hussain alibrahim* | 09/24/2020 |
| Shuyi Zheng | Requirement leader / Front-End Designer 2 | *Shuyi Zheng* | 09/24/2020 |
| Shreyas Prakash | Design and Implementation Leader | *Shreyas Prakash* | 09/24/2020 |
| Neha Jadhav Sarnaik | Design and Implementation Leader | *Neha Jadhav Sarnaik* | 09/24/2020 |
| Ming Yuan | Security Leader | *Ming Yuan* | 09/24/2020 |
| Bhavesh Tadikonda | Backup Team leader | *Bhavesh* | 09/24/2020 |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Change** |
| 1.1 | Neha Jadhav Sarnaik | 10/22/2020 | Introduction,Classes and methods ,Software architecture |
| 1.1 | Shreyas Prakash | 10/22/2020 | Design Patterns , Key Algorithms |

[Introduction](#_87t9hln2vjz0)

[Software Architecture](#_buttcq9i221r)

[Design Patterns](#_x18fj36s1121)

[Key Algorithms](#_mtfbusfb0eq3)

[Classes and Methods](#_7ucksmkf6rzx)

[References](#_15tmymhipvdv)

[Glossary](#_8n34lvocupub)

# Introduction

In this section, give an overview of this document, and also address the design goals of your software system

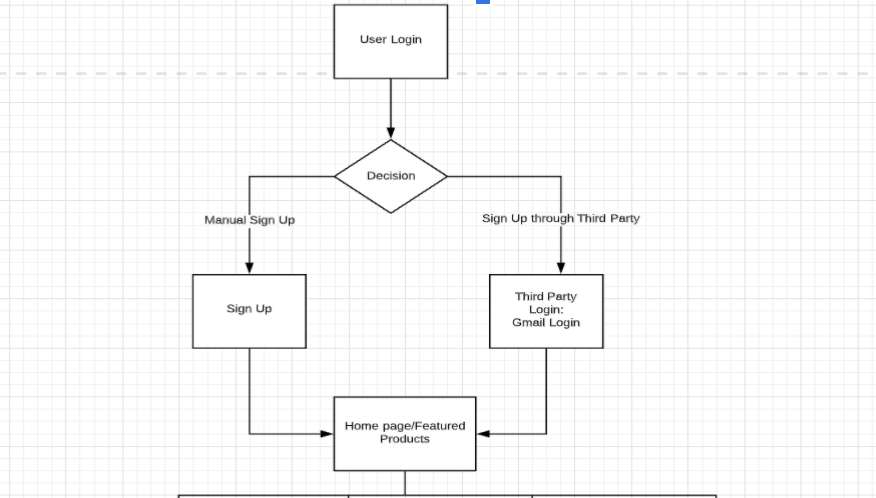
This document contains the total design portrayal of the Moo dairy. This incorporates the architectural features of the framework down through details of what tasks each code module will perform and the database design. It additionally shows how the use cases detailed in the SRS will be executed in the framework utilizing this design. The essential audience of this document are the software developers.The purpose behind this venture is to create a functional site for individuals to have the option to purchase dairy items. Clients of this site will have the option to create an account which will supply them flexibly with a simple to use interface so as to buy products recorded . The basic requirements express that the clients will have the option to securely register manually or through third party login and manage their profiles. They will likewise have the option to explore through available items by category, or by utilizing a well executed search function.

Any member can register and see accessible products. Just registered members can buy numerous products paying regardless of quantity. Users can view and buy products. An Admin has some additional benefits including all benefits of the user. Administrators can include products, edit product data and add/remove products. Administrator can include users, edit user data and can remove users.

# Software Architecture

In this section, you will describe the decomposition of your software system, which include each component (which may be in terms of package or folder) and the relationship between components. You shall have a diagram to show the whole architecture, and class diagram for each component. The interface of each component and dependency between components should also be described. If any framework is used, it shall be defined here too. Database design should also be described if used.

**High level design**



Microservice architecture has been used as the application functionality can be divided into multiple independent modules that can precisely perform standalone tasks.

The modules in the architecture .First we have the account and authorization module and the product module .

Account and Authorization Module

This part makes and records users data in a database with different priority and authority, which may allow customers to have their own account by manually registering or logging in or by signing up or logging in through the third party google login to buy. Related architecture is straight forward. Administrator authorization required.

Product Module

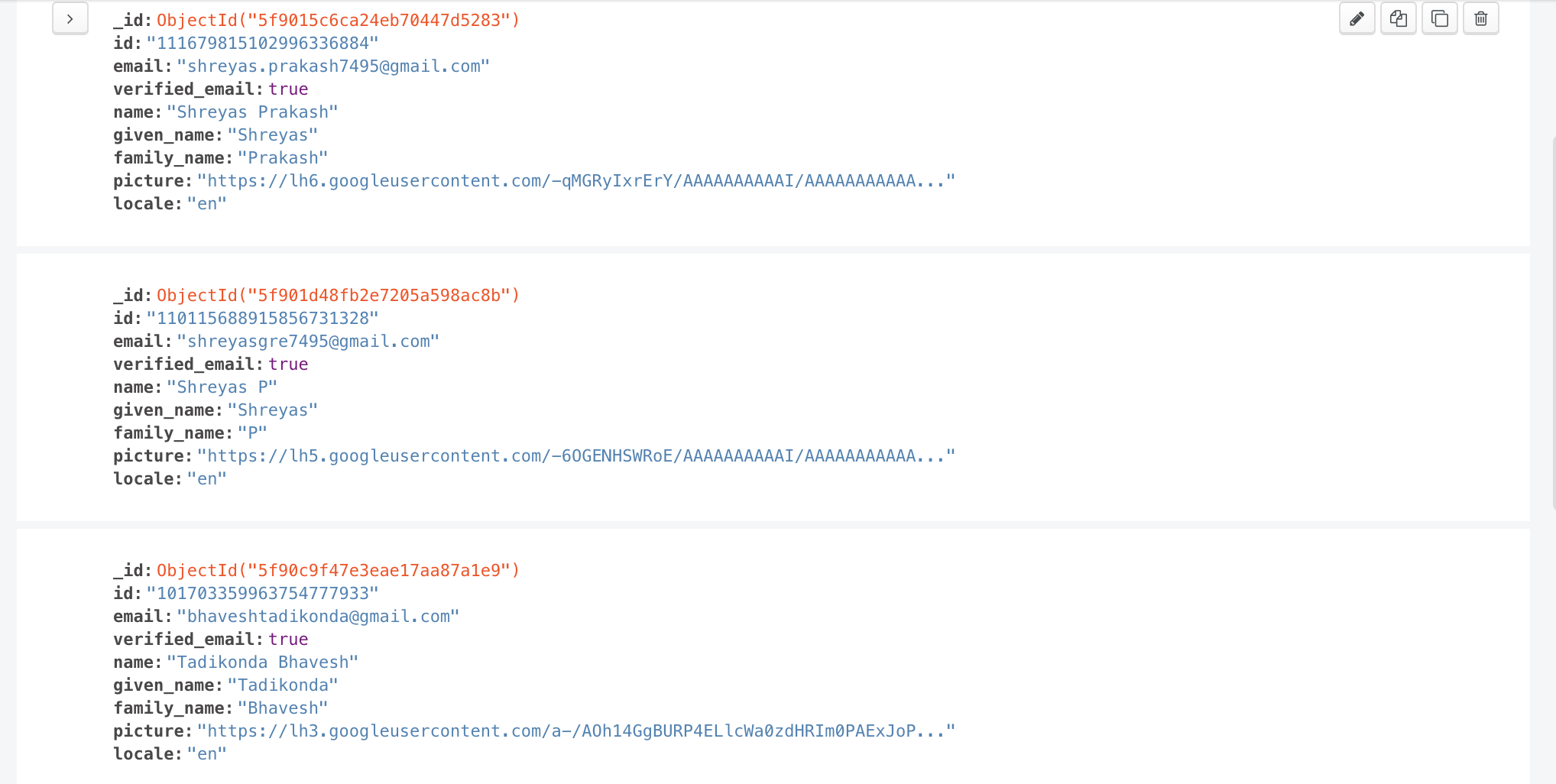
A portion of its functions are selective for customers . This module expects designers to execute browse and search function for our web visitors and should be as amicable as could be expected under the circumstances and as solid as could reasonably be expected (for example quick, 7-24 assistance, barely break down and recovery)

# Database Design (if applied)

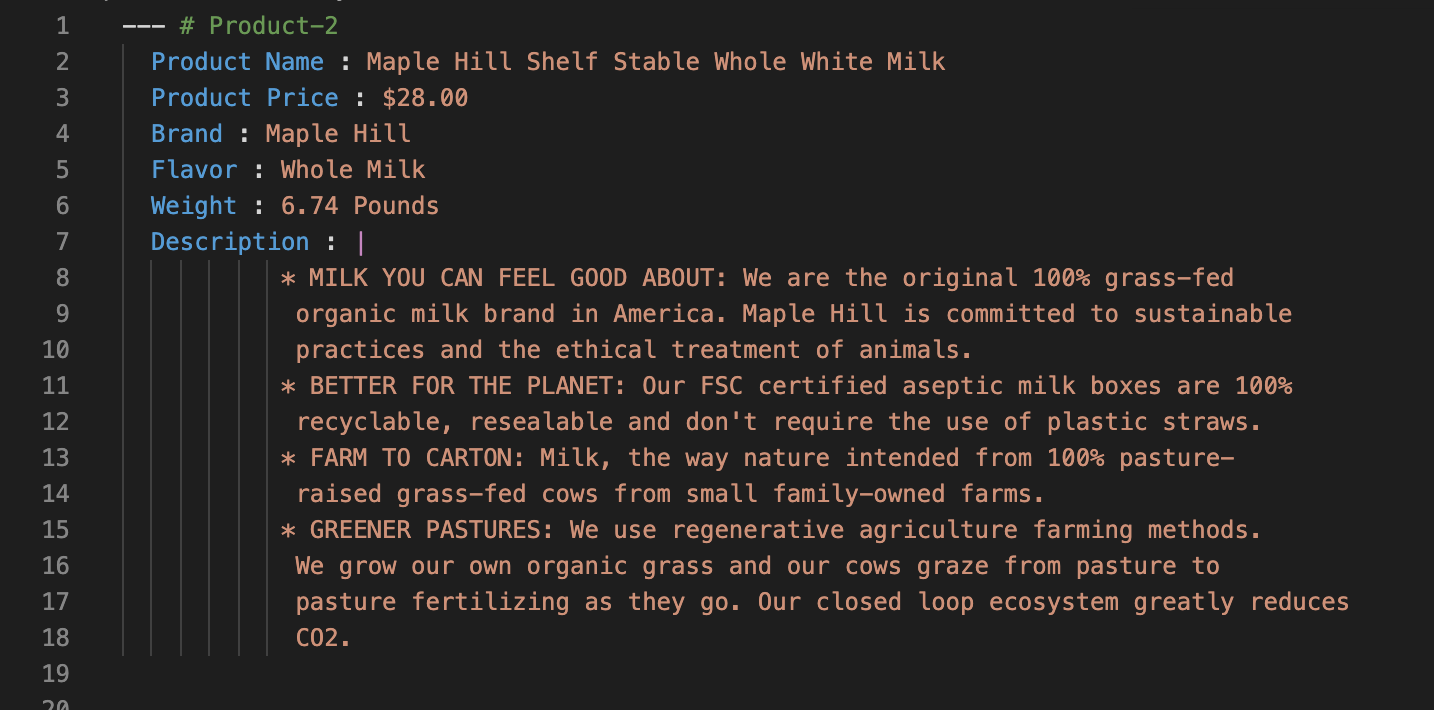
We are using MongoDB, which is a NoSql database. This is a document database, information in documents stored as Json objects. Objects are nested as Key-value pairs.

Here we are performing data sharding based on the functionality results which is Manual Login, Google Login and Product Collections.

1. OAUTH\_USER\_DETAILS: This is a collection which is used to store the User Login details in MongoDB. These details are important considering the fact that we might need the Login Information of the particular user who has accessed the application at a given point of time and date. Also we can get the count of the number of times a particular user has logged In into our application through this collection. Format of data stored on OAUTH\_USER\_DETAILS collection is below:



1. PRODUCT: This is a collection which contains information about all the products that we are using in our application. We are initially creating a YAML config file for each product so that it is easily understandable and can be used later by the QA team in order to compare and check the results easily. We have written a python script that converts a YAML file to JSON and dynamically stores this JSON into MongoDB. Format of YAML used to store product information is as below:

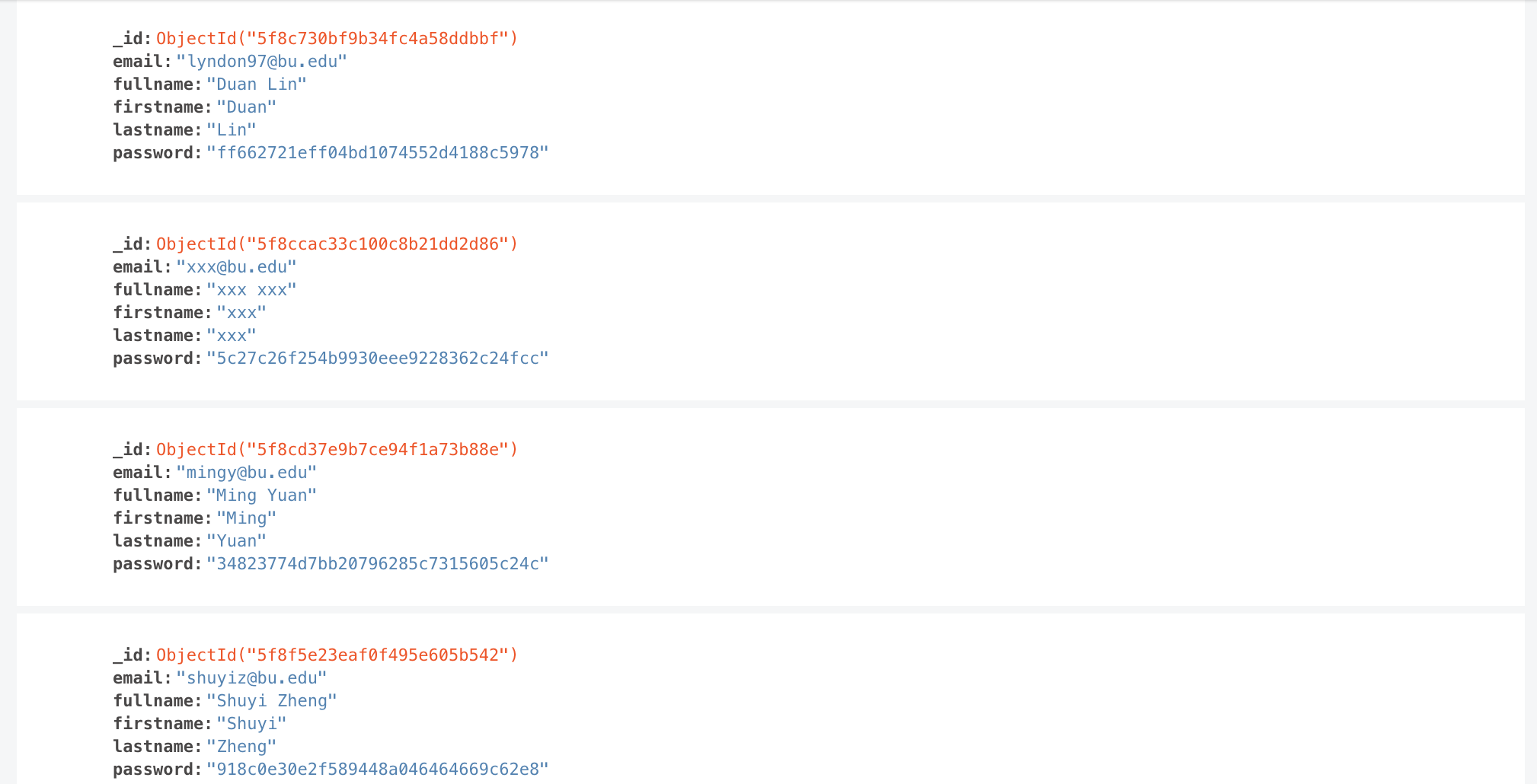


Format of data stored on PRODUCT collection is:



This collection is used to get the product information for the user whenever the user is interested in the product. This collection is dynamically populated with a new product with its product ID whenever we want to add a new product into the product list. Also this collection has to be updated now and then which means that we have to remove the entries of products if they are running out of stock.

1. USER: This collection is used to store user manual Login Details. This collection is used to perform analytics on the data present on such as : To find the number of new users added on a particular day, to find how many users logged In into the application on a given day. Also this collection is used to check if the user has already signed up with the application to give permission to Log In to our application. The format of data stored on USER collection is as follows:



# Design Patterns

In this section, you shall describe any design patterns used in your software system.

Design pattern that we are using for building our application is Singleton. We are defining the functionalities in different classes which are: Class Product , Class Login, Class Cart, Class Payment and Class Checkout. We are using only a single instance of these classes whenever we are rendering the functionalities defined in a particular class.

This is achieved by importing the class definitions and using a single instance of that class to access all the functionalities of that particular class wherever required.

Class Product has all the functionalities related to products which includes product information, Add a product to cart, Delete a product from cart.

Class Login has two different functionalities which includes Manual sign in and Google Sign in.

Class Cart has all the functionalities related to cart which includes Showing all products on Cart, proceed to payment, Update Cart.

Class Payment has all functionalities related to payment which includes pay through credit card, Pay through Amazon Pay, Pay through PayPal, Cancel Payment.

Class Checkout has a functionality to give the status of confirmation to the User.

# Key Algorithms

In this section, you shall describe any key algorithms used in your software system, either in terms of pseudocode or flowchart.

The Key Algorithms used in our System are as follows:

1. Manual Sign Up : User is allowed to do Manual Sign Up for the application.

Pseudocode is as follows:

i) Get the user Information

ii) Check if the user information is already present in the MongoDB.

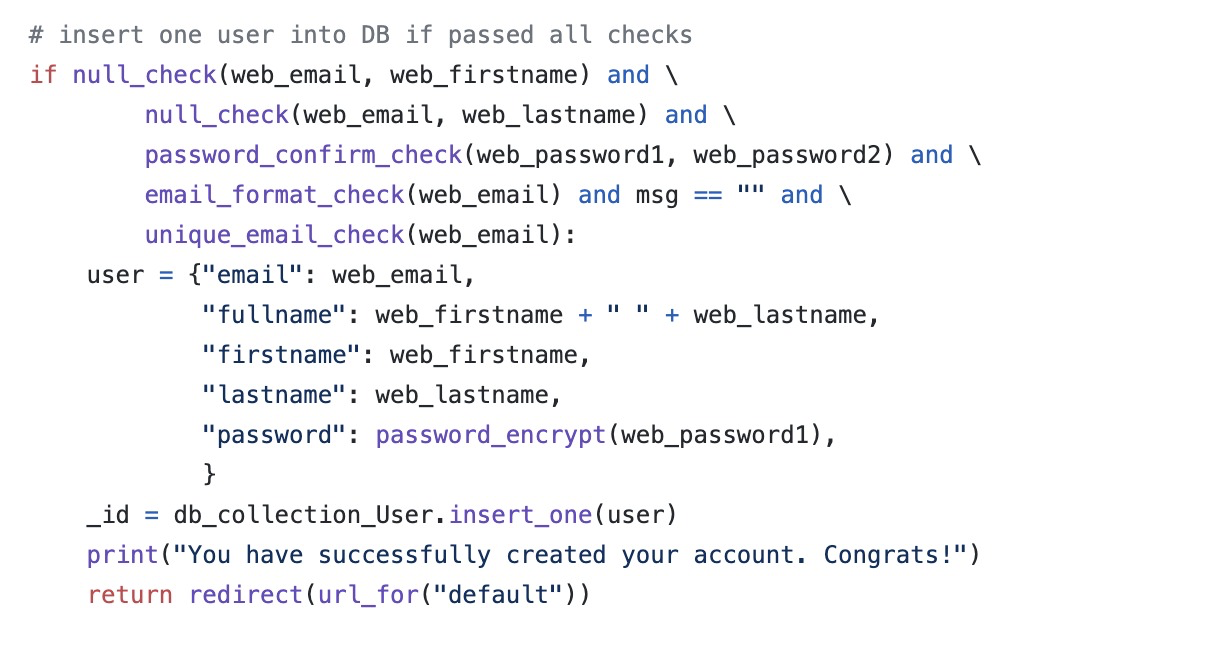
If the user information is already present:

Return “User has an account already, try logging!”.

Else:

Update the information of user on the MongoDB

Return “User can now Log In”



The code above is the structure for Manual Log In.

1. Google Sign In: Users can use third party Sign In in order to access the application and check the products. We have used Google Login Authentication here using OAuth.

Pseudocode is as follows:

i) Upon arrival at the Login page, Choose for Google Log In.

ii) Upon clicking Google Login, it will redirect to ‘/login’ route of app.py where we have written the logic to display gmail ID’s of the users to choose.

iii) User will choose the Email ID

iv) OAuth will verify if the Email ID to proceed for Login:

If the verification is successful:

Redirect to ‘/authorize’ route:

Get the user details and store in JSON

Store the User Details(JSON) to MongoDB(OAUTH\_USER\_DETAILS)

Return to Home Page

Else:

Return “Verification Failed! Please check the Email ID”

The Code Structure for OAuth Google Verification is as below:



1. Password Encryption and Decryption Algorithm: When the users are doing Manual Sign Up, they give the password and these passwords are being securely stored in the database by Encryption and Decryption techniques so that no attacker(Bad Bots attack or DDos attack) will be able to access this information on the Database.

Pseudocode is as follows:

i) Get the Username and password that has to be stored on MongoDB from User.

ii) Create a hashcode by using IP Hashing technique which will give us a unique hashcode for each individual person’s account.

iii) Verify the Email ID credentials by OAuth by making use of Google API.

iv) If Verification is Successful:

Store the credentials successfully on MongoDB.

Else:

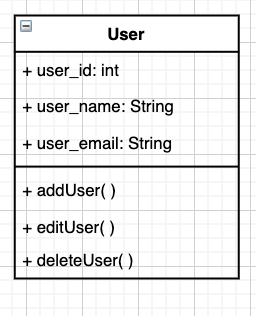
Return “Email Verification Failed! Please try again”

Once we store the IP hashcode that had been generated during the profile creation on MongoDB, from now on only the Admin of the System will have the list of IP Hashcodes which has been generated before Logging In inorder to make sure that Application is secure. This is done so that only Admin or the member of the System who has access to this map of Hashcodes is to User will be able to access the contents of database and will be secured from external attacks such as DDos and Bad Bots by producing this Hashcode when they are trying to access the database. This is when the decryption phenomenon occurs and the access is given only to these members of the system.

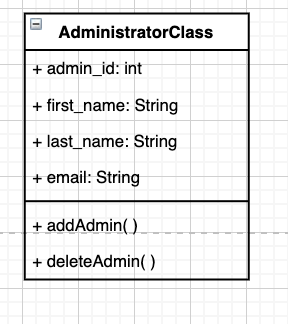
# Classes and Methods

This part can be a reference to an automatic generated document for all classes and methods.

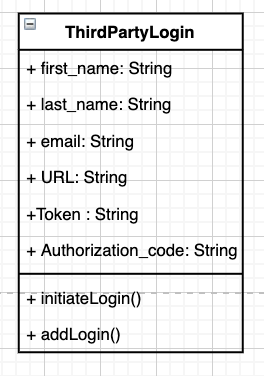
Third party login class diagram



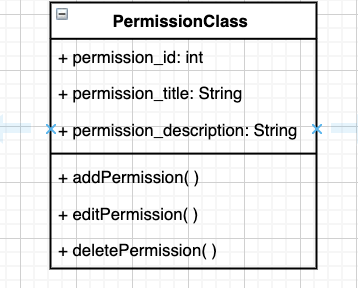
In the user class there are different attributes such as the user\_id , user\_name, user\_email and the methods are the addUser() ,editUser(), deleteUser() which are required for third party or google login . The user class is utilized to store the details of the apparent users. It is utilized to create a new user or help a current user login.



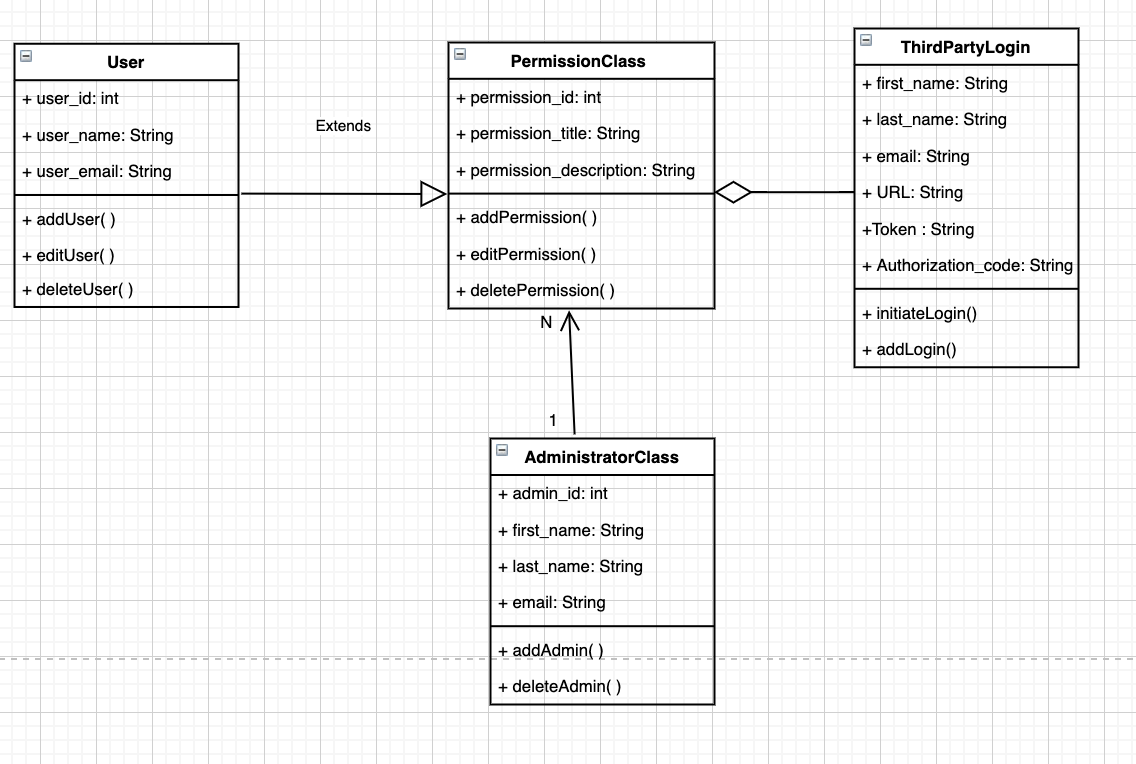
In the administrator class there are different attributes such as admin\_id ,first\_name,last\_name ,email and the methods are addAdmin() , deleteAdmin() .An administrator has a larger number of benefits than an ordinary user. Main elements of an administrator will be the administration of users.Administrators can see, change and delete the personal information and passwords of members if important. They would likewise have the option to search information about members .



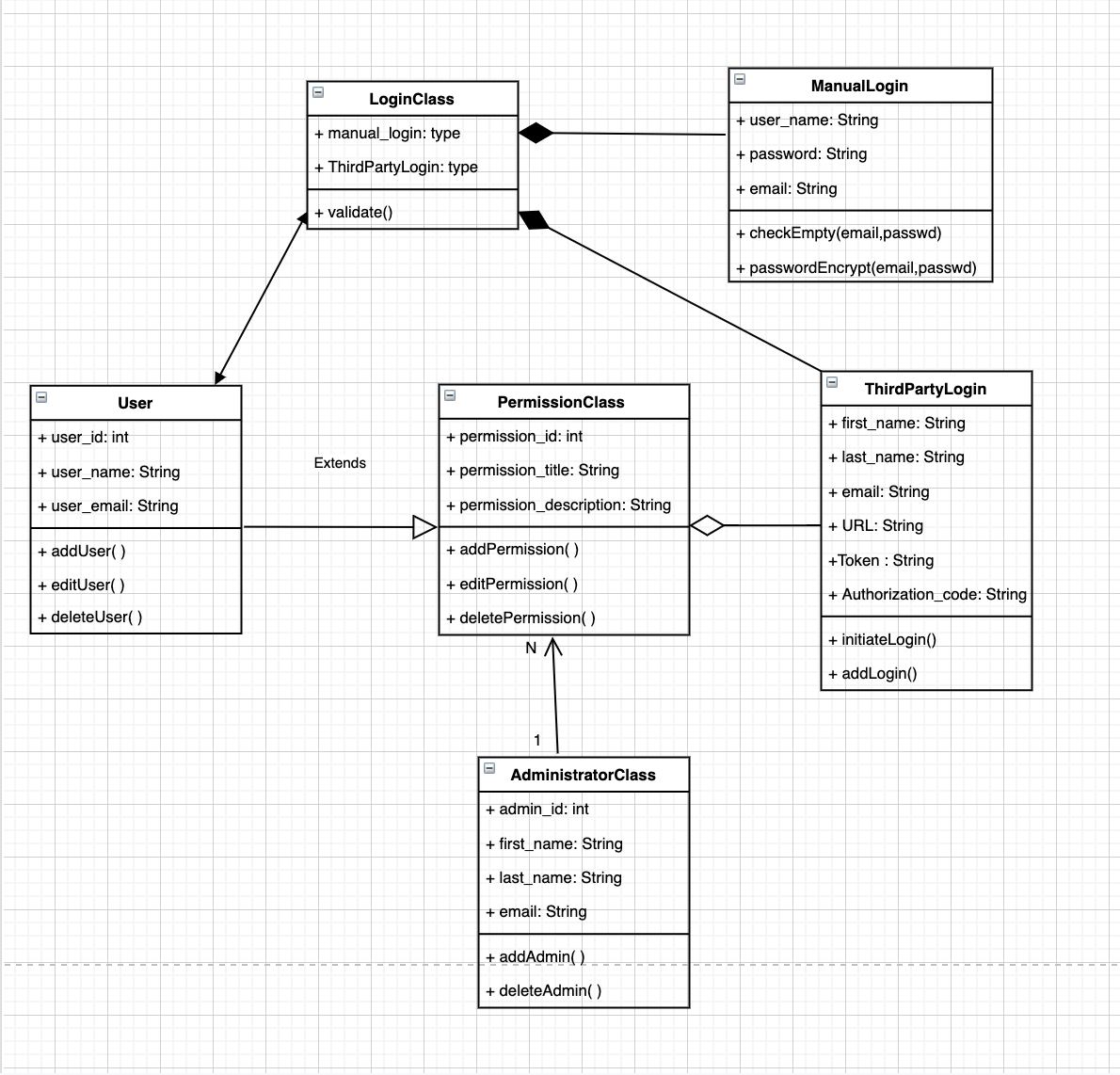
In the ThirdPartyLogin class there are different types of attributes such as the first\_name , last\_name , email, URL ,Token , Authorization\_code and different methods are initiateLogin() and addLogin().



In Permission class there are different types of attributes such as the permission\_id , permission\_title, permission\_description and different methods are addPermission(), editPermission(), deletePermission().



Login class diagram



# References

[1] us dairy <https://www.usdairy.com/>

[2] instacart <https://www.instacart.com/>

# Glossary