UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Bellisimo
Requirements Specification

Vision and Scope

1. Project Background

Bellisimo is a company aimed at providing an online platform for customers to browse and purchase clothing as well as food catalogues provided by the business located in Hatfield. Information about specials and promotions will be published on the online platform.

2. Vision and Scope

The core of the system will be catalogues of items and their prices. Since Bellisimo is involved in clothing and food, the catalogues will have to ensure that these lines are well maintained. Sales and specials in each line will have to be accounted for and managed. The scope of the system is to ensure that the latest information is being provided about items and their prices. Additionally the system has to allow users to purchase the products provided by Bellisimo by adding them to the online carts that are associated with the profiles.

3. Architecture Design of Bellisimo System

The Bellisimo system is a monolithic system that operates as a single unit at the high level, it uses Maven as the dependency management tool. However at a lower level it employs the Model-View-Controller (MVC) architecture. The model is the Spring Boot backend application, the view is the Angular2 application which the user is exposed to and interacts with and the controller is the HTTP calls performed using the REST framework.

Architectural Patterns:

Model-view-control architecture patterns – to allow the decoupling of the frontend from the backend of the system so they can be implemented and maintained independently

Quality Requirements

(a) Performance – The ability to retrieve content as well as update content in a reasonable amount of time.

(c) Availability – The Bellisimo system should be available and accessible whenever a user has a need to use its functionality.

(d) Maintainability – The ease at which the system can be upgraded and enhanced to include additional functionality or remove functionality.

(e) Scalability – The ability to add and remove specials, food and clothing items to the system without affecting the functioning of the system.

(f) Reliability – The degree to which the content received by the user from the system can be trusted.

(g) Security – To ensure that only authenticated and verified users are able to change and remove the content that is displayed on the Bellisimo system and users are able to view what is in their carts and orders list.

4. Design Requirement

The system should be designed in such a way that there are two different interfaces that depict the clothing catalogue and the food catalogue. The following design specifications are what the system should adhere to:

There should be at least two interfaces.

- An admin interface to maintain the catalogues
- A regular user interface that allows people to interactively view the catalogue as well as buy the products they are interested in by adding them to the shopping cart associate with their profile.

5. Technologies

The following technologies will be used to implement the system:

- Html 5 (Html and Bootstrap CSS)
- Angular2
- NodeJS
- Spring Boot
- PostgresSQL
- Apache Maven
- Git (Github)

The web application will consist of two subsystems that communicate via HTTP using REST Framework. The Java/Spring Boot application will be known as the "backend" application. The HTML5/Angular2 application will be known as the "frontend" application. The backend application is expected to communicate with the database and use Hibernate which can be imported into Maven, a dependency management tool whereas the frontend application will

be hosted in the browser and NodeJS is expected to manage packages required for the application to run successfully.

Modules and Responsibilities

6. User Management Module:

6.1 Scope

Responsible for the authentication of users and facilitates the management of the admin user as well as the regular user profiles and their related information. This includes the credentials and the contact details of the administrators. Since the system contains information about various food and clothing outlets the system can accommodate more than one administrator. Additionally the systems has to allow users who are interested in purchasing their products to create user profiles which will store their carts as well as information which will be used when the users eventually order the products.

- Admin User: A registered user who can add and remove website content including specials.
- Regular User: A registered user who can add and remove products from their shopping cart and eventually place an order for the products in their cart.
- Guest User: A user who can only use the system for viewing the content and information of the various food and clothing items

A Guest User is not stored in the database. For an Admin users and regular users the email address has to be unique for each user to avoid duplication of user accounts, additionally the password field of a user is encrypted. This is to enforce the security requirement of the system. The authenticate use case is initiated when the user tries to login into the system using their registered email and password. Additionally the login use case will make use of the getUser service which will only return a user object if the email address and the password used for authentication match a user who is already in the system.

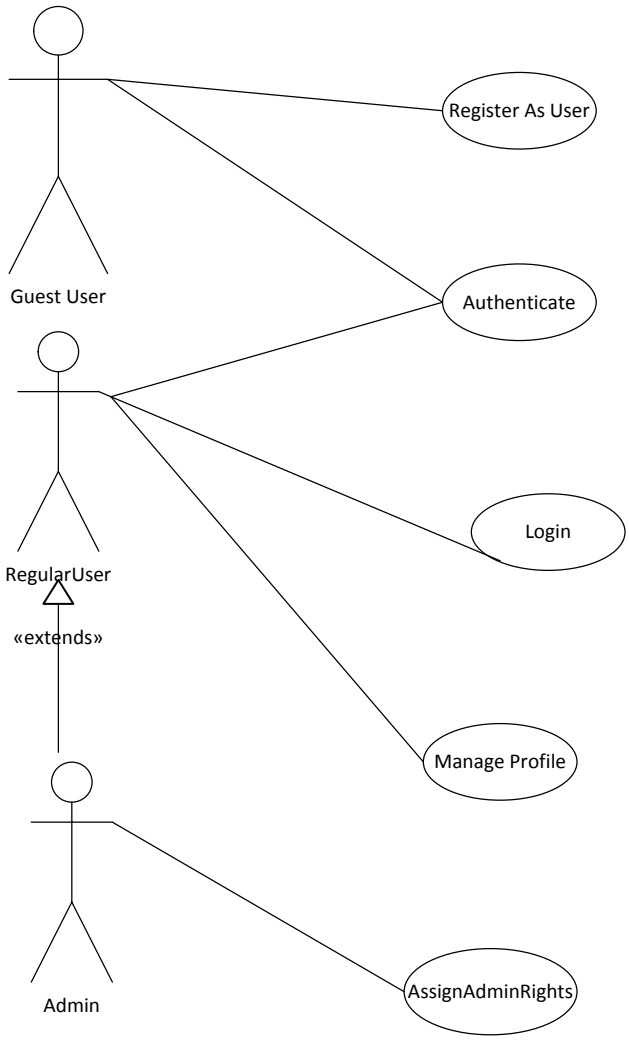


Figure 1: User Management Module - Use Cases

6.2 Domain Model

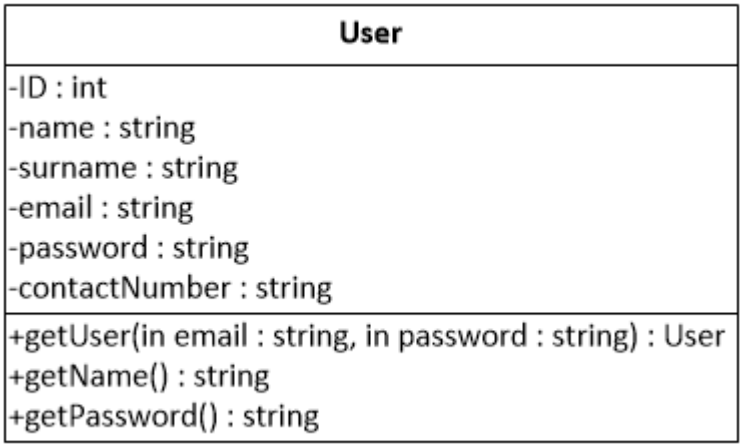


Figure 2: User Management Module -Domain Model

6.3 Services Contracts

The following services contracts are provided:

- getUser: This service returns a registered user from the database if the provided email and password match a user object.
  Pre-condition: A user with the provided email and password is registered in the system.
  Post-Condition: If a user is matched, a user object will be returned else an exception will be thrown. No changes will be made to the database.

6.4 Comments

The Register and manage profile functionality was not implemented.

7. Data Management Module:

7.1 Scope

The data management module is responsible for all the storage, retrieval and maintenance of the information that is required by the system. This includes storing the data that is used by the user management module i.e. the login credentials of the administrators. It also stores information about the products that Bellisimo has to offer. This includes the clothing and food items as well as all the information about specials and promotions that the store may be conducting, what discount is applicable to each special as well as how the long the promotion is valid for. Admin users are responsible for the maintenance of this information. The user data such as their password and email can only be changed in the backend i.e. direct manipulation of the database as a result no functionality is provided for updating user information on the front-end. Product information on the other hand can be manipulated from the front-end of the system by an authenticated and logged in admin user. The admin user can perform the following functions on the products using the related services contract:

- Create product
- Read product
- Update product
- Delete product

The details of these functions will be elaborated in the services contract section of the data management module.

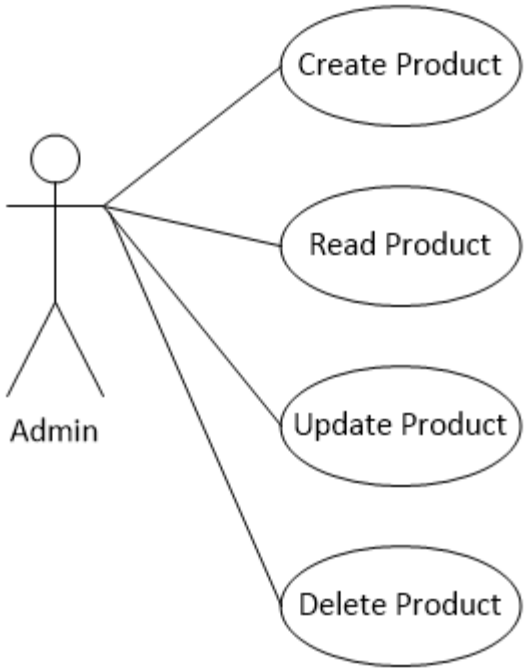


Figure 3: Data Management Module - Use Cases

7.2 Domain Model

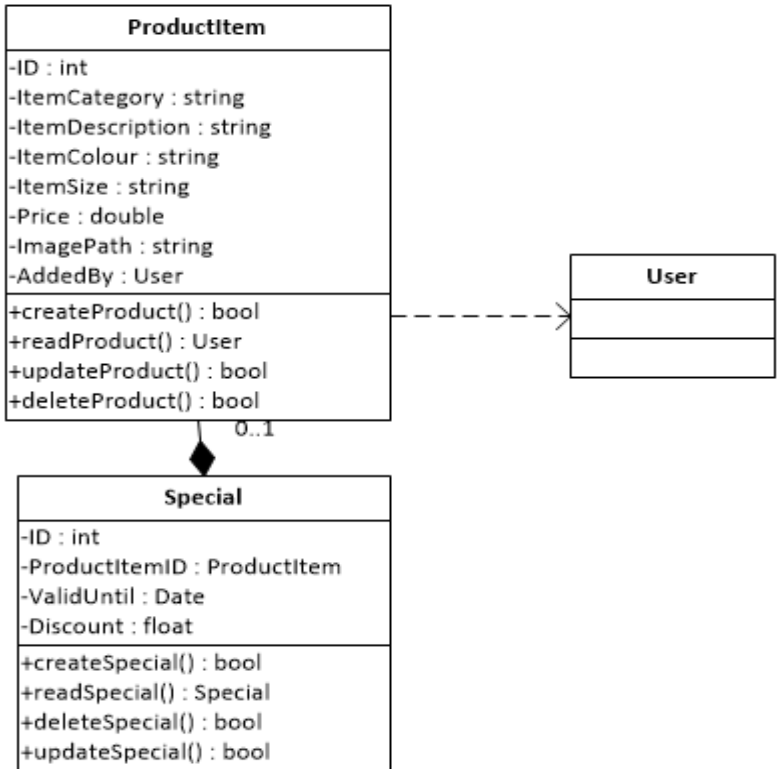


Figure 4: Data Management Module- Domain Model

7.3 Services Contracts

The following services contracts are provided:

- createProduct: This entails adding an item to the system that doesn't exist currently, meaning that if a duplicate item already exists it will not be added to the system. The admin user must have all the information about the item i.e. item type: clothing or food, item price etc. If the item is a promotion then the validity period of the item must be specified
  Pre-condition: The item doesn't exists in the system.
  Post-condition: The item is added to the system if it didn't exist previously, if it did exist an exception is thrown.
- readProduct: Retrieving the details of an existing item.
  Pre-condition: The item has been added to the system
  Post-condition: A product item is returned, if it doesn't exist an exception is thrown.
- updateProduct: This entails updating the details of an existing product item, at least one field of the product must have changed else no change occurs in the system
  Pre-condition: The item exists in the system and atleast one field has to be changed.
  Post-condition: The item has been updated with new field values, if no field has changed an exception is thrown
- deleteProduct: This entails removing a product item from the system, if the item was part of a special, it is removed from that special as well.
  Pre-condition: The product exists in the system.
  Post-condition: The product is removed from the system.

  The updateProduct and the deleteProduct services contract use the readProduct service to retrieve the item to delete of update from the system.

8. View Management Module:

8.1 Scope

The view management model is responsible for all the access related functionality of the system. Specifically the functions that are used by the guest user or the general public. This include browsing the products offered by Bellisimo as well as seeing all the details of the products. It also includes the filter and the search functionality that enhance the user experience.

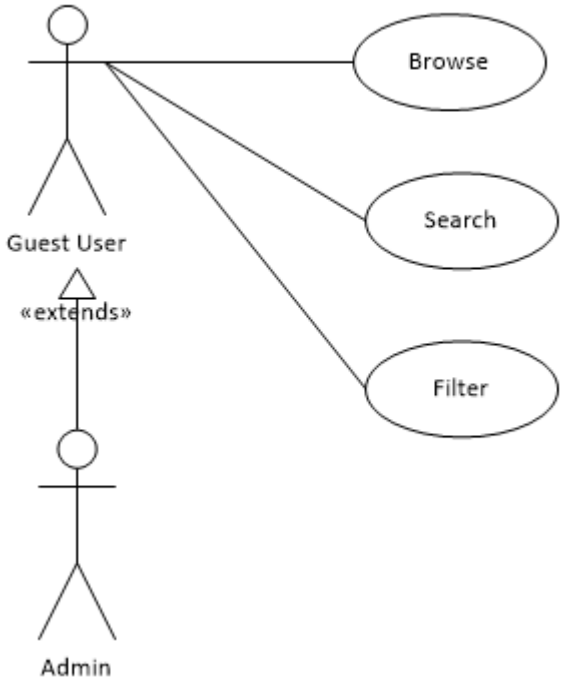


Figure 5: View Management Module - Use Cases

8.2 Services Contracts

The following services contracts are provided:

- Browse: This entails viewing the products that are provided by Bellisimo on the front-end of the system using the clothing and food interface
  Pre-condition: The products have been added to the system by an administrator and the user has accessed the front-end of the system.
  Post-condition: The products are returned via the respective interfaces.
- Filter: This entails refining the products which are returned for browsing to only display products within a certain range or domain e.g. Only show items less than R100
  Pre-condition: The browse service contract has been invoked and the user is browsing the products
  Post-condition: A subset of the browse results are returned within the user's specified range.
- Search: This entails returning only the product items that match a certain criteria e.g. Colour red (not implemented)
  Pre-condition: The browse service contract has been invoked and the user is browsing the products
  Post-condition: A subset of the browse results are returned that matched the specified criteria.

9. Ordering Management Module:

9.1 Scope

The ordering management module deals with all functionality related to the purchasing of the products offered by Bellisimo. This includes the adding and removing of the products from the cart associated with the user profile, updating and viewing order details, checking out an order and the banks validation of orders. The system will use mocking to simulate the role of the bank for electronic payment the only method of payment when doing online shopping with Bellisimo. The mock is setup in such a way that the bank will either accept and honour the payment or decline the payment at a ratio of accept: decline = 95:05. If accepted, the transaction is persisted. If declined, the transaction is cancelled. The ordering interface of the system is only visible to registered users i.e. Regular users and Admin users.

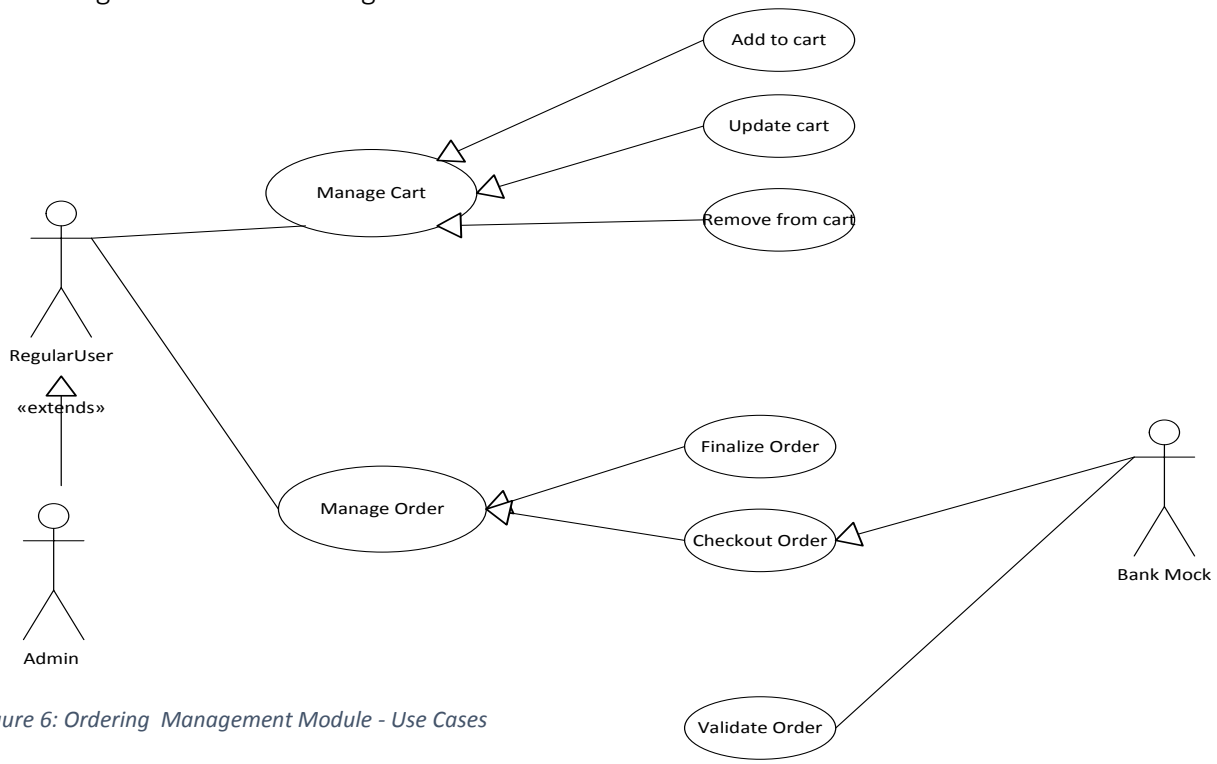


Figure 6: Ordering Management Module - Use Cases

9.2 Services Contracts

The following services contracts are provided:

- Add to cart: This entails selecting a product from the system using the getProduct service contract from the data module and adding it to the cart that is associated with the user profile

  Pre-condition: The products have been added to the system by an administrator and the user has accessed the ordering interface of the system after they have created a profile and logged in.

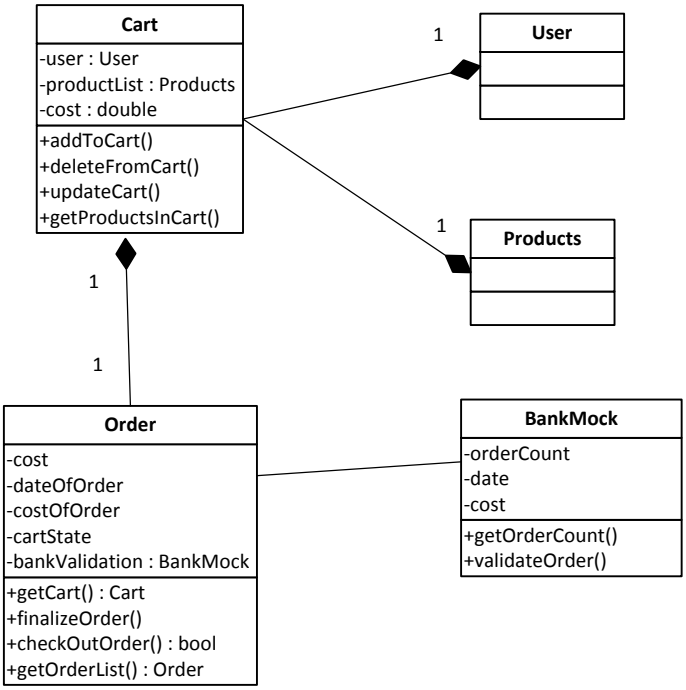
  Post-condition: The products are added to the user's cart and the total cost of the cart increases.
- Remove from cart: This entails deleting a product from the user's cart.

  Pre-condition: The product is removed from the cart and the total cost of the cart decreases.

Post-condition: The products are added to the user's cart.

- Update cart: This entails updating the quantity of the products added to the cart
  Pre-condition: The products are already added to the user's cart
  Post-condition: The products quantity and total cost of the cart are updated
- Checkout cart: This entails selecting the products in the cart for purchase and inserting the order details such as address and contact details.
  Pre-condition: The user has added products to the cart
  Post-condition: The user's cart is emptied and the finalize order service contract is invoked.
- Finalize order: This entails using the mock electronic payment to validate whether an order is declined or accepted
  Pre-condition: The order has been completed and relevant information such as the cost have been determined
  Post-condition: The bank will reply with either a decline or accept status
- View ordering history: This entails viewing all the orders a user has placed on the system since the creation of their profile.
  Pre-condition: The user has created / registered a profile on the system.
  Post-condition: The list of order from the beginning of time is returned to the user.

7.3 Domain Model



74. Comments

The Bank Mock functionality was not implemented.