Project Case Study - Cake Shop (Caketeers)

**Problem Statement**

"Caketeers" is an e-commerce initiative based on a cake shop. Cakes are a constantly expanding industry that is aiming to enter the e-commerce market. Caketeers allow users to search for and purchase numerous cakes in the store based on multiple categories.

The visitor will be directed to a user-friendly website where he or she can browse a variety of cake selections or customize a cake to order directly from the store. The user must be logged in to place an order, as well as to follow order history, amend details, and so on.

**Functional Requirements**

Below details will be focused on the following modules:

**Registration Module Details**

* Title
* First Name
* Last Name
* Email Id
* Address Details (First Line, City, Country, Postcode)
* Contact Number
* Password
* Confirm Password
* Verify as human (optional)

**Login Module Details**

* User email
* Password
* Forgot Password

**Searching Module Details**

* Category search (Anniversary, Birthday, Celebration, Wedding)
* Types of Cake Search
* Product List Module

**Ordering Module Details**

* Navigate through options
* Select cake
* Notes on Cake
* Add to cart
* Cart Module

**Custom Order Details**

* List of options to choose
* Add to Cart
* Cart Module

**Checkout Cart Module**

* Confirm Details
* Place Order

**Review Module Details**

* Customer
* Product
* Rating
* Review
* Submit

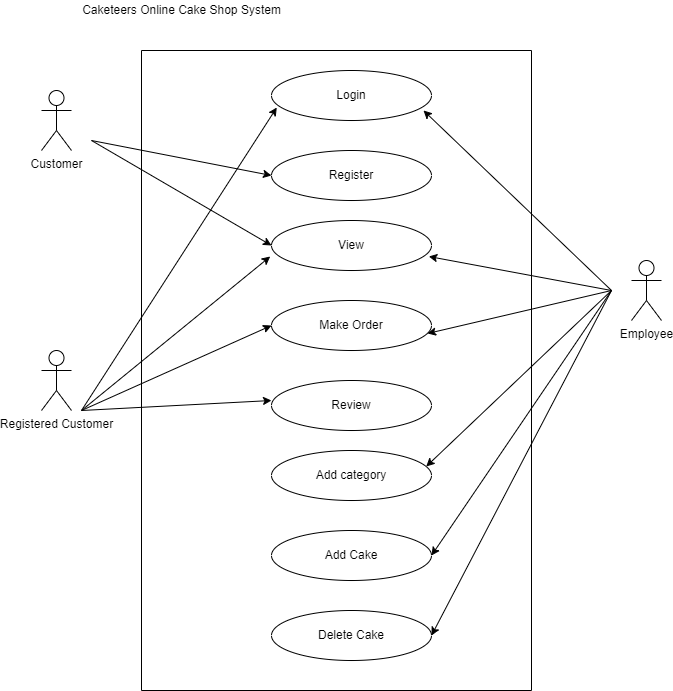
**Product View Module**

* All products listed
* Product Images
* Product Description
* Reviews
* Rating
* Prices

**Profile Module**

* User Details
* Option to update details
* Can see previous and current orders

**Use Case Diagram**



**Page Description**

**Home Page**

This is the default screen when application loads, users can begin to explore through and view various products. Though, they would need to register and login to add products to cart and submit an order.

**Login Page**

The login page will contain the login parameters like username, password, forgot password. The user has to be logged in to order the cake.

**Registration Page**

If the user is not registered, he/she can register by filling in a short form and can then continue to order the products, after logging in.

**Cart Page**

The page contains all the selected products to order and once the user wishes to checkout, the page will ask the user to confirm the delivery details and place the order.

**Reviews Page**

Each product can be reviewed and rated by the customers for the other users to see.

**Products Master Page**

This page contains all the products (cakes), with all the relevant details related to each product and an option to add to cart.

**Custom Order Page**

This page provides the user with the chance to order a cake as per his/her requirements by choosing the options from drop down menu and once completed and add the customized product to the cart.

**Category Page:**

Category page shows lots of categories and types of cakes available. Customers can select a category and view all cakes with that category, making it easier for them to find the cake they want.

**About us Page**

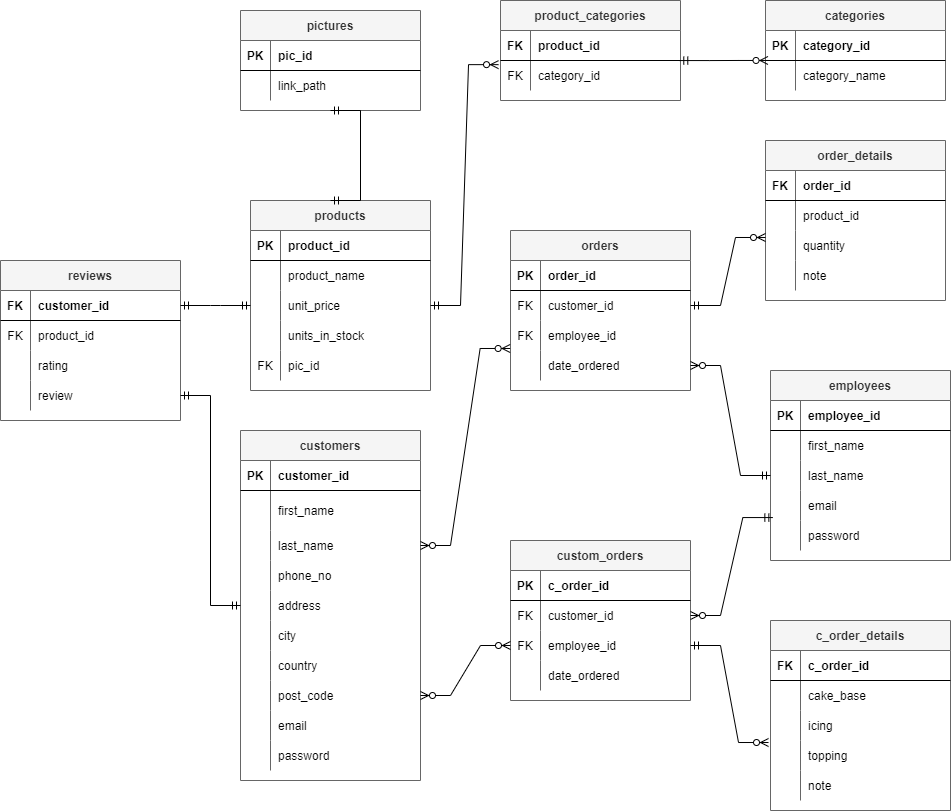
Visitors and Registered users can contact website owners or administrators from this page. They can also get the basic information about the shop.

**Mystery Cake Page:**

This is an advanced and fun page, where users can choose to get any random cake from any category/type and place order.

**Database Design (ER Diagram)**

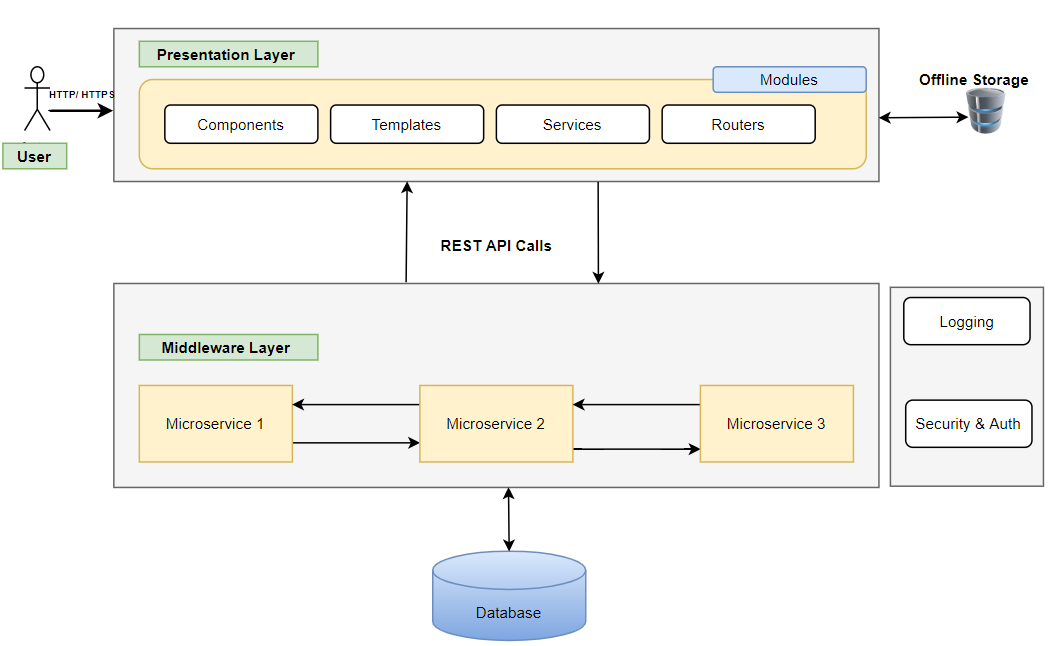
The diagram below shows the database tables used in this case study.



**Non-Functional Requirements**

* The application UI should be supported on Google Chrome and Microsoft Edge
* All the coding guidelines and code quality standards to be followed
* Proper database schema’s to be created
* Follow best practices for databases
* Unit Testing code coverage should be 80%
* Unit Testing for UI is optional
* The application should follow proper naming conventions for both front-end as well as backend codes
* The application should be completed with 80% working functionalities to be accepted for final presentation

**Application Architecture**



Component Description

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Component** | **Description** | **Technology Stack** |
| 1 | Presentation Layer | User Interface (UI) will be developed using HTML5, CSS3, Bootstrap, and Angular 10 JavaScript Framework. These application components will be responsible for rendering User visualization elements, UI processing, data binding, event wiring and command dispatching. | * HTML5, CSS3 * Angular 10+ * Bootstrap or Angular Material for Responsive Design |
| 2 | Offline Storage | Browser has the ability to store the data offline | * Optional (Browser-based or offline storage) |
| 3 | Middleware Layer | Microservices are a collection of services which represent business capabilities. And are highly maintainable, testable, loosely coupled and independently deployable. | * Spring Boot * Swagger API for documentation or any other tool * Spring REST API * Spring Boot Data JPA |
| 4 | Database | Database to the relation and maintain entities data in the tables. Retrieve | * MySQL/ Postgres SQL |
| 5 | Cross Cutting | **Logging** – Useful logs can provide the developer (especially when someone has to debug/maintain someone else’s code) with tremendous help when trying to understand what the code actually does.    **Authentication** - The services would be secured by username/ password or Token based authentication. Authorization can also be performed at API service-level. | * In-built Logging APIs * Integrated Tools/ DataDOG/ LogRocket * NLOG/Log4J/Logback * Spring Security * JWT |

Project Plan Milestones

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Milestones** | **Deliverables** | **Actors** | **Estimated Date for Demonstration** |
| **1** | M1 | * Infrastructure Setup * Project Team Finalization * Selection of Case Study * Project Flow Documentation * Git Repository Folder Structure | Trainer + Dev Team |  |
| **2** | M2 | * Design Wireframes for decided modules * Define JSON structure for UI development of Modules * Complete UI functionality for Login, Logout and landing Dashboard | Dev Team |  |
| **3** | M3 | * Develop UI Screens for other modules <<name here>> * Develop UI Screen for Admin module | Dev Team |  |
| **4** | M4 | * Design Database Schema * Create ER Diagrams * Create other objects like procedures and functions * Finalize the database tables | Dev Team |  |
| **5** | M5 | * Develop Backend REST APIs * Create REST API for Login and Registration * Integrate with UI * Generate Swagger Documentation * Integrate with UI | Dev Team |  |
| **6** | M6 | * Develop remaining backend microservices * Develop REST APIs to perform REST calls * Generate Swagger Documentation * Integrate with UI * Perform server-side validations * Add Logging and Security | Dev Team |  |
| **7** | M7 | * Integrate all Modules | Dev Team +  Trainer |  |

**Definition of Done (DOD)**

* Completed activity is demonstration ready
* All the codes are at least unit tested
* Modules should pass all the validations (UI and Business layers)
* At the end of every milestone, a demo is given to the stakeholders
* Project team will have minimum 3 members and all have contributed in the development
* Every member should be presenting during demonstrations
* Few assumptions can be made while developing the project. However, it needs to be discussed with the trainer