# Problem Statement: Toy shop (Upasana and Lukas)

E-commerce or electronic commerce refers to a business model that involves sales transactions being done. Virtually every online shopping website follows this structure. Any website where you can obtain items for sale over the internet is considered an E-Commerce Website.

The objective of this project is to use to buy toys for Kids. Users can Register and easily search for the requirement from the database. Those users who have registered can log in and add their desired items to the shopping cart.

A web application where users can shop according to their requirements. In this, we have catalogs for toys and have categories of products according to age and choices. Registered users can feel free to add their items to the cart and immediately after placing an order an email notification will be sent to the user as well as a generated invoice.

# Functional Requirements

**Registration Module Details**

For registration details below are to be captured:

* Title
* First Name
* Last Name
* Email Id
* Date of birth
* Address Details
* Contact Number
* Password
* Confirm Password

The system must have validators to ensure the correct form of email address.

The system must validate *password* equals to *confirm password*.

The system must ensure all values are present and fields not empty when submitting.

**Login Module Details**

The system must receive the following details to be captured:

* User email
* Password

**Categories Module Details**

For categories details:

* Action Hero
* Soft Toys
* Puzzles
* Video Games
* Science
* Age 0-6
* Age 6-12
* Age 12+

The system must sort the products by the product tags which are acquired from the object values in the database.

**Shopping Cart Module Details**

* Must let users add.
* Must contain a thumbnail of the image.
* Must let users set quantity (set to 0 to remove).

**Product Module Details**

When creating a new product these details must be captured and added to the database:

* ID
* Name
* Supplier Name
* Price
* Units in Stock
* Discontinued
* Tags [ ]

**Invoice Module Details**

The system must allow the file to be downloadable.

The system must store the file in the database.

The invoice must be a .txt file.

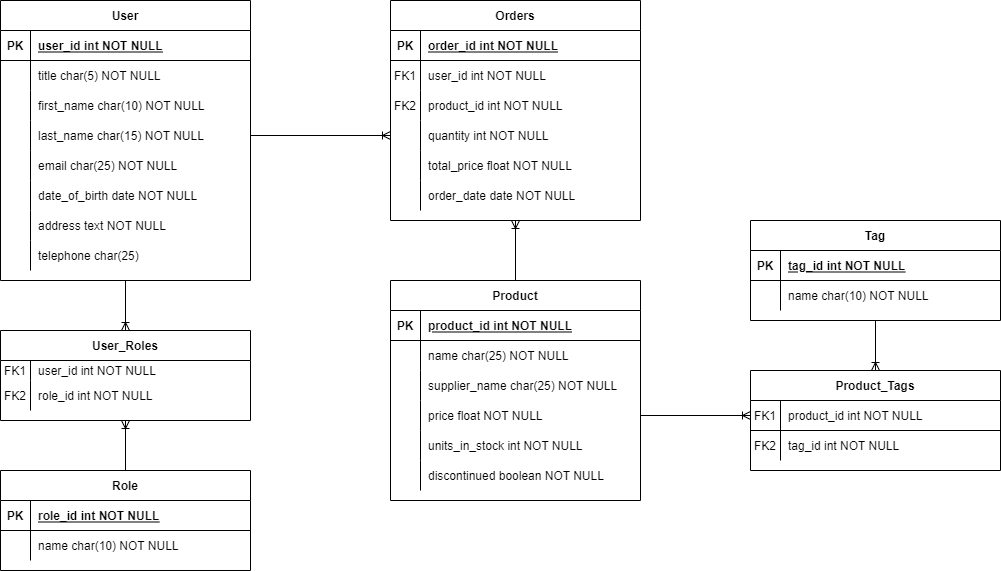
The invoice must contain the following details:

* Product ID
* User ID
* Date Purchased

### **Use Case Diagram**



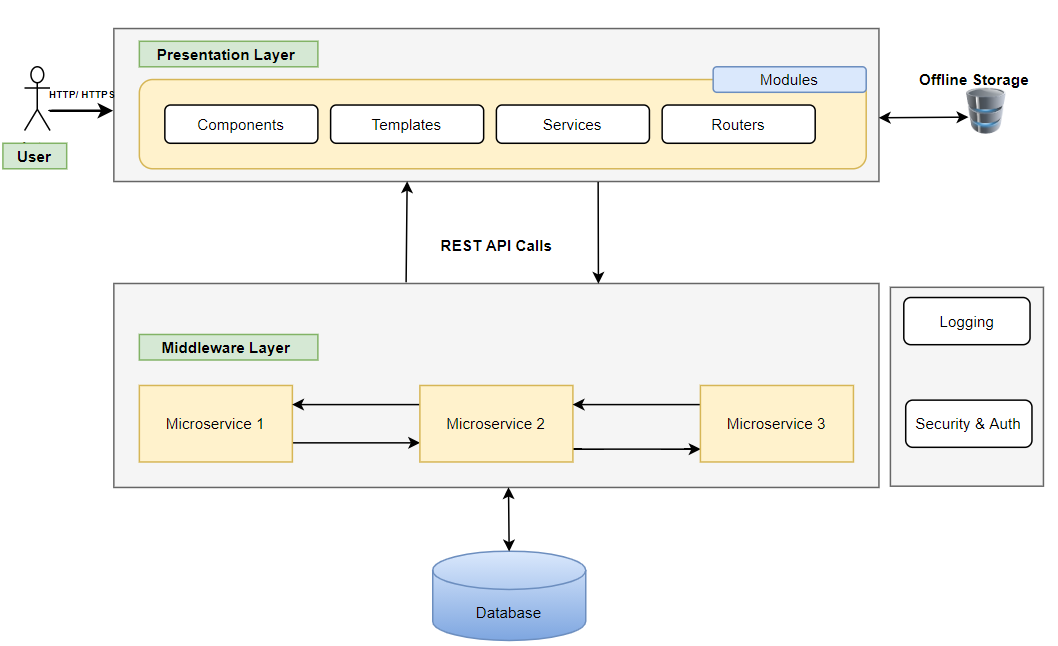
### **Database Design**



# Non-Functional Requirements

* Activities within the web application should be simple and consistent. Navigating throughout the system should be done through concise links and buttons.
* The system should perform its required functions under stated conditions for an extended period of time. Even if the web application has been open for a long time it should still function as required and the data should still be fetched and not corrupt.
* The application should not have any special performance constraints. Within the context of the system, it is expected to execute operations within 10 seconds depending on the internet and machine speed.
* The application should be built and analyzed in a way that would allow/handle any new or future improvements.
* The web application should be available to be used by every user through the internet at any given time. Even if the performance of the system is lacking the application should still be available for viewing and navigation.
* 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 ability to store the data offline | * Optional (Browser-based or offline storage) |
| 3 | Middleware Layer | Microservices are 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 stackholders
* 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 done while developing the project. However, it needs to be discussed with the trainer