# Assumptions

Some requirement is not clear so I make some assumptions to clarify the ambiguity.

**Navigation**

When the user visit the system, the system will display Dashbaord page.

When the user has login, the system will redirect the user to My Profile page.

Breadcrumb is not clickable.

**Login**

Login page header does not show current time, so will not display current time in the login page header.

If login failed, display error message in the login page.

The system does not have sign up page, so the system is not able to register new user. There is some existing user in the system, only these users are able to login into the system.

**Forget Password**

If username not exist, display error message.

If username exist, tell user to check their email for the temporary password.

If the user forgot their username, no other way to recovery their account.

The backend api will generate a random password, every time the password is different.

Url ‘ForgetPwsd’ should be ‘ForgetPswd’.

**My Profile**

The requirement does not mention user email is a link, assume it has a mailto link.

**Change Password**

If the old password is incorrect, frontend display error message.

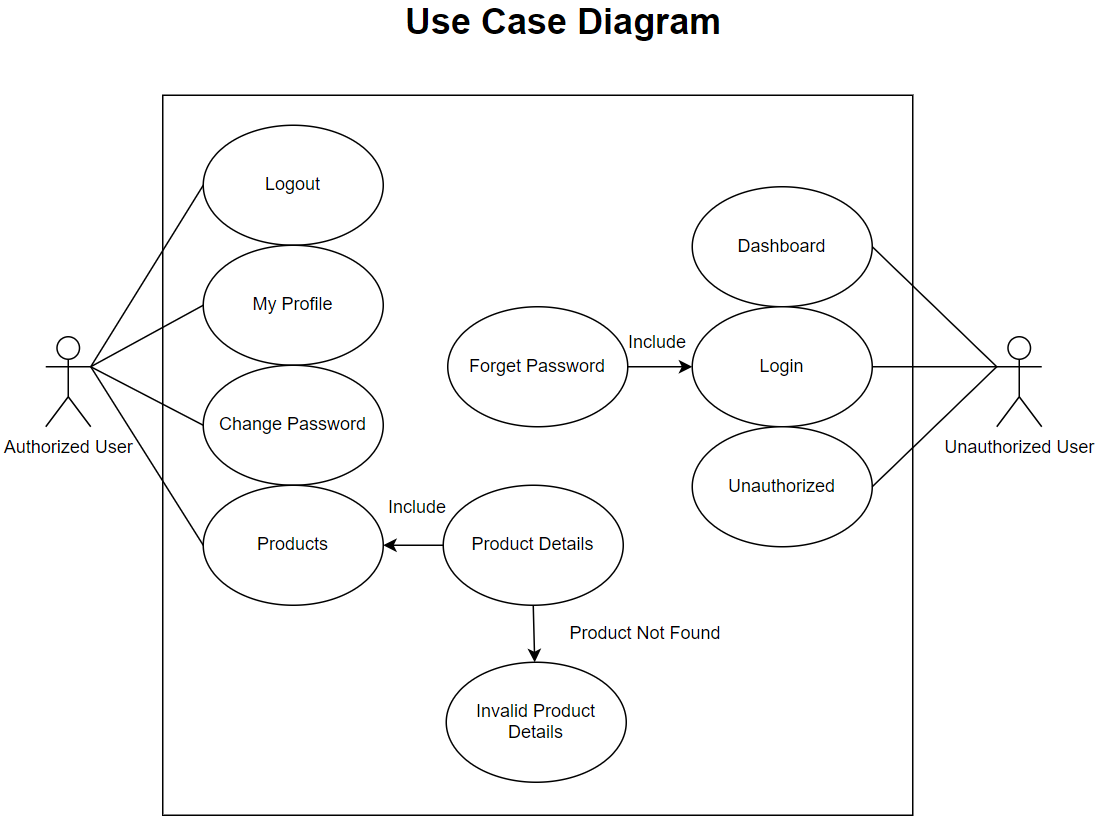
If the old password is correct, tell user the password has change successfully.

The requirement does not mention password needs to be strong, but will verify the password is strong or not for security.

Strong password criteria: minimum eight characters, at least one uppercase letter, one lowercase letter, one number and one special character.

Url ‘ChangePwsd’ should be ‘ChangePswd’

# Use Case Diagram



# Technology Used

Frontend: Angular

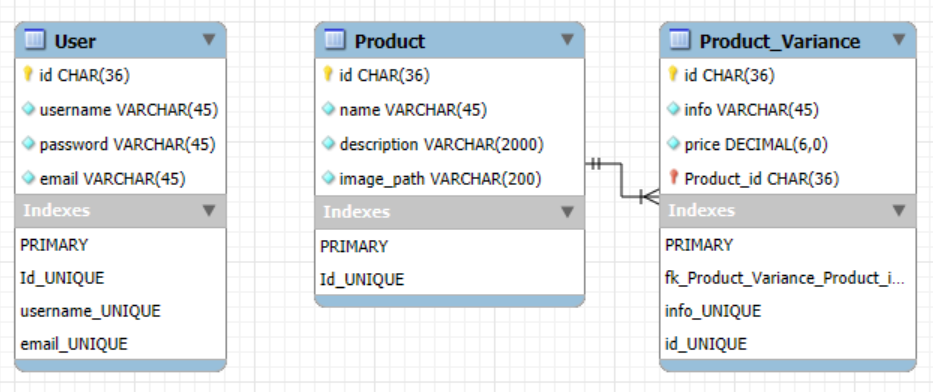
Backend: .NET

Database: MySQL

Database first approach

# Backend

## Database Design



**User**

Id: char(36) (PK)

Username: varchar(45) (not null, unique)

Password: varchar(45) (not null)

Email: varchar(45) (not null, unique)

**Product**

Id: char(36) (PK)

Name: varchar(45) (not null)

Description: varchar(2000) (not null)

image\_path: varchar(200) (not null)

\* Product picture will be retrieved from internet.

**Product\_Variance**

Id: char(36) (PK)

Info: varchar(45) (not null, unique)

Price: decimal(6,0) (not null)

Product\_id: char(36) (FK) (not null)

## API Design

Frontend URL:

Backend URL: https://localhost:7046/api

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Feature** | **Frontend Route** | **API end point** | **API Description** | **Request Body** | **Response Body** |
| Home | /Dashboard | - |  | - | - |
| Login | /Login | POST /user/login | Check the username and password are correct. | {  username: string,  password: string  } | If username and password match:  {  user\_id: string,  username: string  }  else {  "status": 404,  Error: Not Found  "message": "The username or password is wrong."  } |
| Forget Password | /ForgetPswd | POST /user/forgetpassword | Send temporary password to user's email. | {  username: string  } | If user name was found  {  Status: 204  }  Else if user name was found but unable to send email  {  Status: 500  Error: internal server error  Message: Username [username] was found but the server was unable to send the temporary password to user’s email.  }  Else if user name was no found  {  Status: 404  Error: Not Found  Message: ‘username [username] not found’  } |
| My Profile | /MyProfile | GET /user/[id] | Retrieve user info. | - | If user id was found  {  username: string,  email: string  }  Else  {  "status": 404,  Error: Not Found  "message": "user with id [id] no found."  } |
| Change Password | /ChangePswd | POST /user/changepassword | Verify current password and change to new password. | {  user\_id: string,  current\_password: string,  new\_password: string  } | If current password is correct and new password is valid  {  Status: 204  }  Else if current password is incorrect  {  Status: 400  Error: bad request  Message: ‘Incorrect password.’  }  Else if current password is correct but new password is invalid  {  Status: 400  Error: bad request  Message: ‘New password is invalid.’  } |
| Products | /Product | GET /product | Retrieve list of products and find price range of each product | - | {  {  id: string,  name: string,  description: string,  image\_path: string,  lower\_price\_range: number,  upper\_price\_range: number  },  {  id: string,  name: string,  description: string,  image\_path: string,  lower\_price\_range: number,  upper\_price\_range: number  },  {  id: string,  name: string,  description: string,  image\_path: string,  lower\_price\_range: number,  upper\_price\_range: number  }  } |
| Product Details  Invalid Product Details | /Product/[id] | GET /product/[id] | Retrieve product details by id. | - | If product was found  {  Id: string  Name: string  Description: string  image\_path: string  product\_variance\_list: {  {  Id: string  Info: string  Price: number  },  {  Id: string  Info: string  Price: number  },  }  }  else  {  "status": 404,  "error": "Not Found",  "message": "product with id [id] no found."  } |
| Unauthorized | /Unauthorized | - |  | - | - |

## Class Diagram

**User**

Id: Guid (PK)

Username: string (not null, unique, max 45 char)

Password: string (not null, max 45 char)

Email: string (not null, email format, max 45 char)

**Product**

Id: Guid (PK)

Name: string (max 45 char, not null)

Description: string (max 2000 char, not null)

image\_path: string (max 200 char, not null)

variance: List<ProductVariance>

**ProductVariance**

Id: Guid (PK)

Info: string (max 45 char, not null)

Price: int (max 6 decimal, not null)

## Data Transfer Object Design

**LoginRequestDTO**

Username

Password

**LoginResponseDTO**

User\_id

Username

**AccountRecoveryRequestDTO**

Username

**MyProfileDTO**

Username

email

**ChangePasswordRequestDTO**

User\_id

Current\_password

New\_password

**ProductDTO**

Id: string

Name: string

Description: string

image\_path: string

lower\_price\_range: number

upper\_price\_range: number

**ProductDetailsDTO**

Id: string

Name: string

Description: string

image\_path: string

product\_variance: ProductVarianceDTO[]

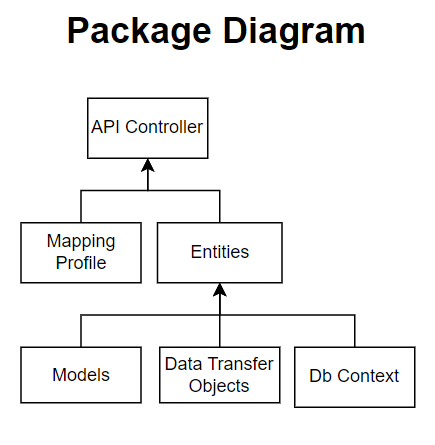
**ProductVarianceDTO** (inside ProductDetailsDTO class)

Id: sting

Info: string

Price: number

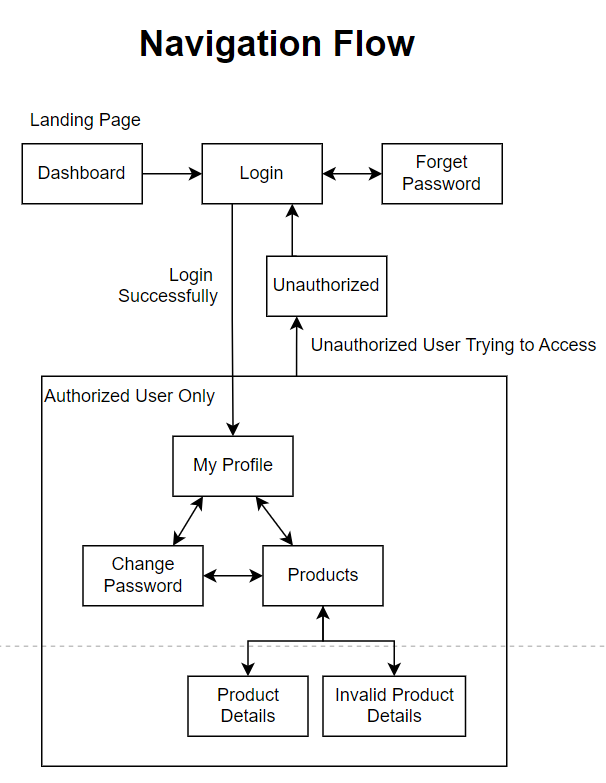
## Backend Package Diagram



\* I did not implement repository pattern because it is just a simple application, using repository pattern might be over design.

# Frontend

## Navigation Flow



## Routes

|  |  |  |
| --- | --- | --- |
| **Feature** | **Frontend Route** | **Component** |
| Navigation | - | Header, CurrentTime |
| - | LeftMenu |
| - | Breadcrumb |
|  | / | /Dashboard |
| Home | /Dashboard | Dashbaord |
| Login | /Login | LoginForm |
| Forget Password | /ForgetPswd | ForgetPassword |
| My Profile | /MyProfile | MyProfile |
| Change Password | /ChangePswd | ChangePasswordForm |
| Products | /Product | ProductList |
| Product Details  Invalid Product Details | /Product/[id] | ProductDetails |
| Unauthorized | /Unauthorized | Unauthorized |

# Effort

Validate input from frontend

Database limit the password to maximum 45 character, also implement input validation in front end.

Use dto to avoid circular reference in json

# Future Improvement

* Hash the password in database.
* Download the product image and save them in an online drive. So no need to get the image from external site we don’t have control, reduce risks.
* Add a custom validation to validate uniqueness in model, rather than validate the input in the controller.
* Implement send temporary password to user’s email for Forget Password feature.