**Document of Project**

**Introduction:**

ShopBridge is an application where the user can track the items in the inventory for sale.

Admin can view the products, list of different products available in stock, list of categories, subcategories, brands, packings. Admin can perform all crud operations. He can search for a particular product.

For example let MI Note 5 mobile is a product. Here :

Category: Electronics

SubCategory: Mobiles

Brand: MI Note 5

Packing: weight of product.

Stock describes the list of different types of products available and the quantity, price, batchcode of each product.

Product is having fields like – productid, name, description, status-whether product is available in stock or not, image of product, MRP, category, subcategory, packing(weight), brand.

**Data Base Layer**:

A Database is a collection of data that is organized so that it can be easily accesses, managed and updated.

In this project, Created the Data Base called ShopBridgeDB which Consists of six tables.

* Category
* SubCategory
* Brand
* Packing
* Product
* Stock

**Data of Tables**

* **Category Table :**
* This table is used to represent the details of all the categories.
* Fields of this table:
* CategoryId (Primary Key)
* CategoryName
* CreatedAt
* CreatedBy
* ModofiedAt
* ModifiedBy
* **SubCategory Table :**
* This table is used to represent the details of all the subcategories.
* Fields of this table:
* SubCategoryId (Primary Key)
* SubCategoryName
* CreatedAt
* CreatedBy
* ModofiedAt
* ModifiedBy
* **Brand Table :**
* This table is used to represent the details of all the brands.
* Fields of this table:
* BrandId (Primary Key)
* BrandName
* CreatedAt
* CreatedBy
* ModofiedAt
* ModifiedBy
* **Packing Table :**
* This table is used to represent the details of all the packings.
* Fields of this table:
* PackingId (Primary Key)
* PackingName
* CreatedAt
* CreatedBy
* ModofiedAt
* ModifiedBy
* **Product Table :**
* This table is used to represent the details of all the products.
* Fields of this table:
* ProductId (Primary Key)
* ProductName
* Image
* MRP
* Description
* CategoryId (Foreign Key)
* SubCategoryId (Foreign Key)
* PackingId (Foreign Key)
* BrandId (Foreign Key)
* Status
* CreatedAt
* CreatedBy
* ModofiedAt
* ModifiedBy
* **Stock Table :**
* This table is used to represent the details of all the stocks.
* Fields of this table:
* StockId (Primary Key)
* ProductId (Foreign Key)
* Quantity
* Purchase Price
* Selling Price
* BatchCode
* CreatedAt
* CreatedBy
* ModofiedAt
* ModifiedBy

**DATA ACCESS LAYER** :

ENTITY FRAMEWORK

* In this layer, By using Entity Frame Work we develop the required application. Entity frame Work uses Databases to link the interfaces.
* It consists of interfaces, by using interfaces we develop the Operations by using class. Class consists of interface method implementations.
* Product Operations :
* By Implementing this interface in the Product Operations class, we can Perform Create, Retrieve, Update and Delete the details of the Product.
* Interface :

public interface IProductOperations

{

Task AddProduct(Product product);

Task<List<Product>> getAllProducts();

Task<Product> getProductById(int id);

Task UpdateProduct(int id, Product product);

Task DeleteProduct(int id);

Task<List<Product>> getAllProductsByCategory(int catId);

Task<List<Product>> getAllProductsByBrand(int brandId);

Task<List<Product>> getAllProductsBySubCategory(int subCatId);

Task<List<Product>> getAllProductsByPacking(int packingId);

Task<List<Product>> searchProducts(string search\_string);

}

* Stock Operations :
* By Implementing this interface in the Stock Operations class, we can Perform Create, Retrieve, Update and Delete the details of the Stock.
* Interface :

public interface IStockOperations

{

Task AddStock(Stock stock);

Task<List<Stock>> getAllStocks();

Task<List<Stock>> getAllStocksByProduct(int productId);

Task<Stock> getStockByStockId(int stockId);

Task UpdateStock(int id, Stock stock);

Task DeleteStock(int stockId);

}

* Category Operations :
* By Implementing this interface in the Category Operations class, we can Perform Create, Retrieve, Update and Delete the details of the Category.
* Interface :

public interface ICategoryOperations

{

Task AddCategory(Category cat);

Task<List<Category>> getAllCategories();

Task<Category> getCategoryById(int id);

Task UpdateCategory(int id, Category cat);

Task DeleteCategory(int id);

}

* SubCategory Operations :
* By Implementing this interface in the SubCategory Operations class, we can Perform Create, Retrieve, Update and Delete the details of the SubCategory.
* Interface :

public interface ISubCategoryOperations

{

Task AddSubCategory(SubCategory subcat);

Task<List<SubCategory>> getAllSubCategories();

Task<SubCategory> getSubCategoryById(int id);

Task UpdateSubCategory(int id, SubCategory subcat);

Task DeleteSubCategory(int id);

}

* Brand Operations :
* By Implementing this interface in the brand Operations class, we can Perform Create, Retrieve, Update and Delete the details of the Brand.
* Interface :

public interface IBrandOpertions

{

Task AddBrand(Brand brand);

Task<List<Brand>> getAllBrands();

Task<Brand> getBrandById(int id);

Task UpdateBrand(int id, Brand brand);

Task DeleteBrand(int id);

}

* Packing Operations :
* By Implementing this interface in the Packing Operations class, we can Perform Create, Retrieve, Update and Delete the details of the Packing.
* Interface :

public interface IPackingOperations

{

Task AddPacking(Packing pack);

Task<List<Packing>> getAllPackings();

Task<Packing> getPackingById(int id);

Task UpdatePacking(int id, Packing pack);

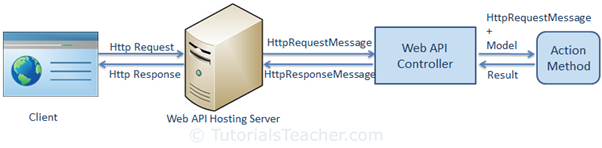
Task DeletePacking(int id);

}

**Web Api Controllers:**

The ASP.NET Web API allows for displaying the data in various formats, such as XML and JSON. It is a framework that uses the HTTP services and makes it easy to provide the response to the client request. The response depends on the request of the clients. The web API builds the HTTP services and manages the request using the HTTP protocols. The Web API is an open source and it can be hosted in the application or on the IIS .The request may be GET, POST, DELETE or PUT. We can say that the WebApi is used for creating restful services. A Rest service supports all http methods like GET,POST,PUT and DELETE.It can be accessed only by specifying a url and http method, but it supports only HTTP protocol.

**WebApi Controller Mechanism**



**CONTROLLERS:-**

Web API controller is a class which can be created under the Controllers folder. The name of a controller class must end with "Controller" and it must be derived from System.Web.Http.ApiController class. All the public methods of the controller are called action methods.

Sample Code:

public class ValuesController : ApiController

{

// GET: api/student

public IEnumerable<string> Get()

{ return new string[] { "value1", "value2" };}

// GET: api/student/5

public string Get(int id) { return "value"; }

// POST: api/student

public void Post([FromBody]string value) { }

// PUT: api/student/5

public void Put(int id, [FromBody]string value) { }

// DELETE: api/student/5

public void Delete(int id) { }

}

Web API routing is similar to ASP.NET MVC Routing. It routes an incoming HTTP request to a particular action method on a Web API controller.

**In IJP Project there are 7 Web Api Controllers**

While using Web API, we have to import the respective library to get operations done. Configuration is made between the controllers and Library Operations to get the data.

A picture containing text, person, screenshot

Description automatically generated

1. BrandController:- This controller will perform all the operations on brand table and return the output in json or xml format
2. Category Controller:-

This controller will perform all the operations on category table and return the output in json or xml format

1. Packing Controller:-

This controller will perform all the operations on packing table and return the output in json or xml format

1. Product Controller:-

This controller will perform all the operations on product table and return the output in json or xml format

1. Stock Controller:-

This controller will perform all the operations on stock table and return the output in json or xml format.

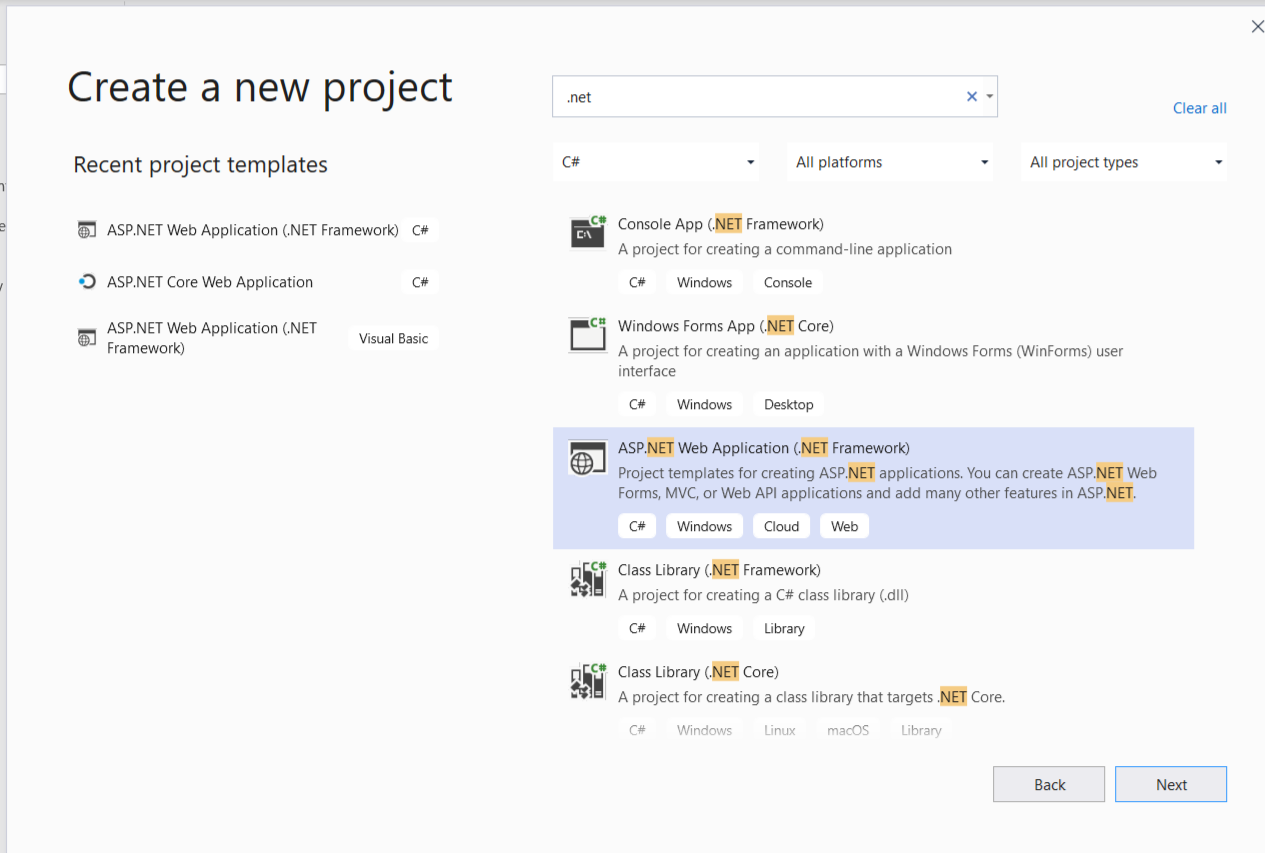
1. SubCategory Controller:-

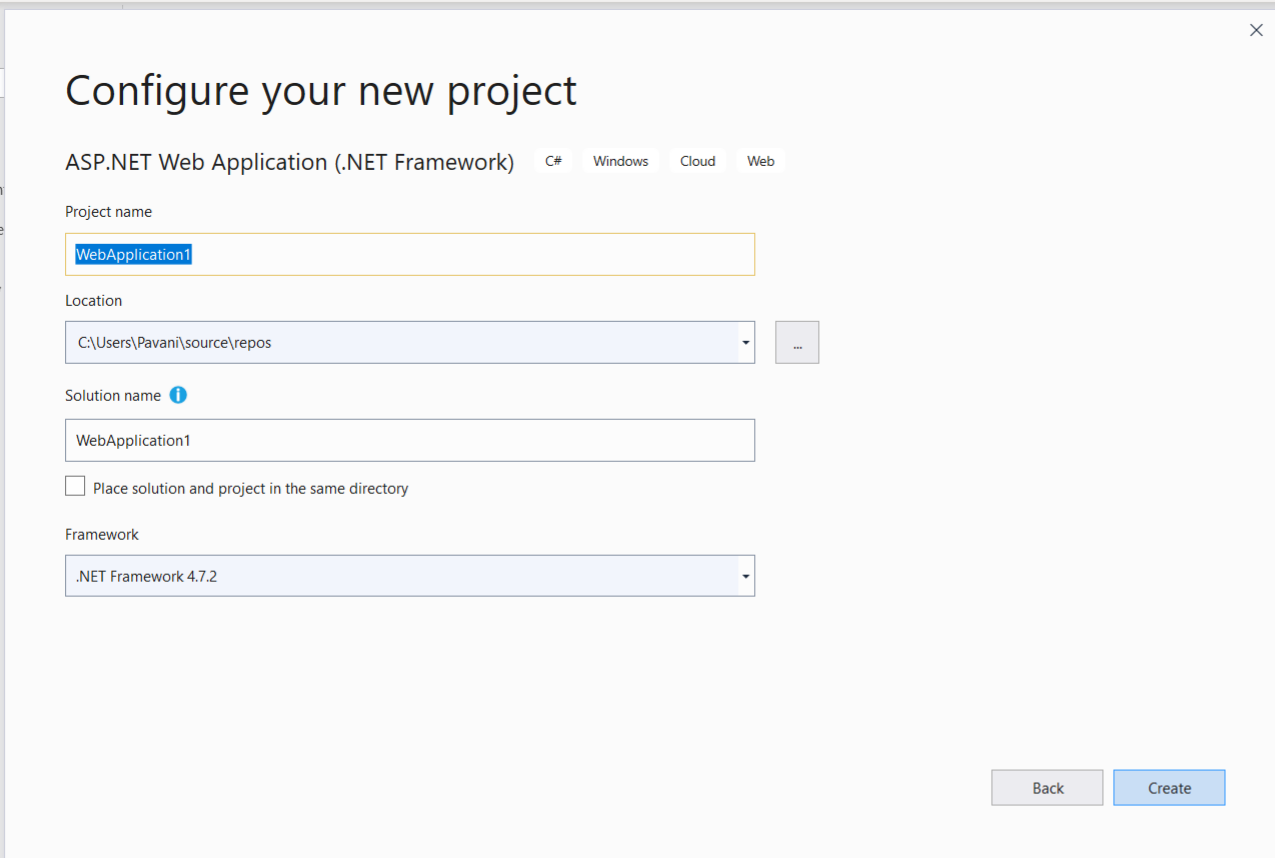
This controller will perform all the operations on subcategory table and return the output in json or xml format

**Procedure:**

1.Open Visual Studio 2019

2.Create a new Project by selecting ASP.NET Web Application(.NET Framework).Then click on next.





3.Give the appropriate project name. Choose the location where you want to locate

your project. Click on create, you will be asked to select a template.

4. A screenshot of a social media post

Description automatically generated

Select Web API template. Then click on create. Project gets created.

A screenshot of a computer screen

Description automatically generated

5. Right click on solution file and Add New Project-Class Library(.Net Framework)

6. Give the database connection to the application.

7. Right-click on Class Library->Add->New Item->ADO.NET Entity Model

A screenshot of a social media post

Description automatically generated

Give the name and click on Add.

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

Give the server name and select your database name and click on OK.

A screenshot of a computer screen

Description automatically generated

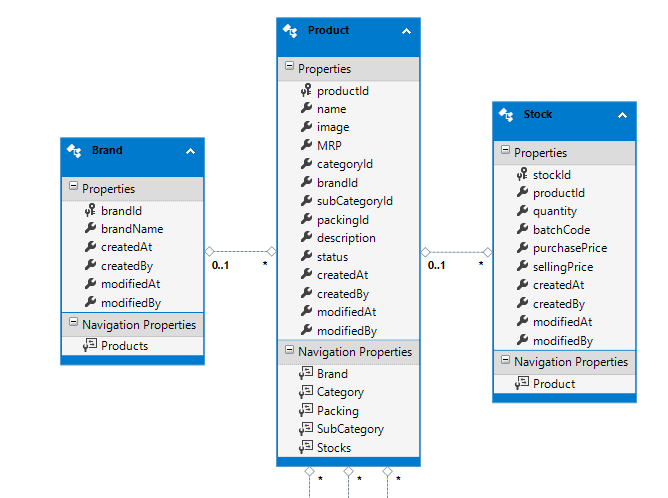
Click on next.

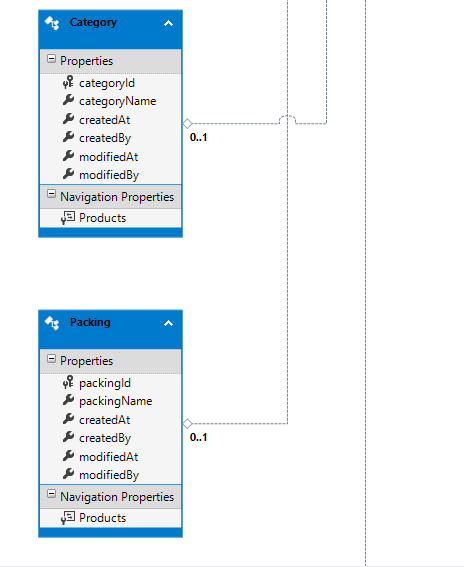
A screenshot of a social media post

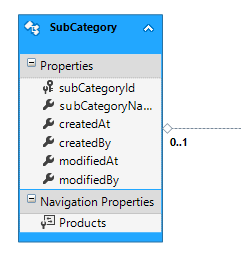
Description automatically generated

Select Tables->dbo->Select the tables->click on Finish.

8.An edmx diagram gets generated in models folder which is a database diagram.







9..Right click on Controller and add Web API Controller with read/write actions.

A screenshot of a cell phone

Description automatically generated