Intro & File Names

1. This System Design Document establishes a clear and comprehensive blueprint for implementing our team's application, with a focus on the "For You Page" and its recipe recommendation workflow. The document details the design level class diagram, including explicit attribute data types, initial values, valid ranges where applicable, and method signatures with parameters and return types, ensuring the "For You Page" class and related entities are precisely defined for consistent implementation. Complementing the static structure, the statechart diagrams capture each object's lifecycle, clarifying how state transitions are triggered by events. Finally, the sequence diagrams articulate end to end interactions for key use cases, browsing the feed, generating personalized recommendations, searching recipes, and updating likes and saves, showing object collaboration, message order, and control flow. Together, these sections provide an actionable, shared understanding of the system's architecture and behavior.
2. DesignClassDiagramProject.vsdx
3. StateChartDiagram.vsdx
4. FCSD1-8.vsdx

Pseudocode

Pseudocode:

Class:

```
CLASS Profile
    PRIVATE attributes:
        profileID: String
        profilename: String
        email: String
        password: String
        fullName: String
        displayName: String
        status: String
        title: String
        photo: String

    // Login Method
    PUBLIC METHOD login(): Boolean
        inputEmail = GET_USER_INPUT("Email")
        inputPass = GET_USER_INPUT("Password")

        IF inputEmail == this.email AND inputPass == this.password THEN
            this.status = "Active"
            RETURN True
        ELSE
            RETURN False
        END IF
    END METHOD

    // Update Profile Information
    PUBLIC METHOD updateProfile(): Void
        this.displayName = GET_USER_INPUT("Display Name")
        this.title = GET_USER_INPUT("Title")
        saveProfile()
```

```
    END METHOD

    // Save changes to database
    PUBLIC METHOD saveProfile(): Boolean
        IF DATABASE.UPDATE("Profiles", this.profileID, this) THEN
            RETURN True
        ELSE
            RETURN False
        END IF
    END METHOD

    PUBLIC METHOD uploadPhoto(): Void
        newImage = FILE_SELECTOR()
        IF newImage.isValid() THEN
            this.photo = newImage.path
        END IF
    END METHOD
END CLASS
```

Shopping List:

```
CLASS ShoppingList
    PRIVATE attributes:
        listID: String
        items: List<Item>
        quantity: Float
        totalPrice: Float

    // Add an item object to the list
    PUBLIC METHOD addItem(newItem: Item): Void
        this.items.ADD(newItem)
        calculateTotal()
```

```
    END METHOD

    // Remove an item object
    PUBLIC METHOD removeItem(item: Item): Void
        IF this.items.CONTAINS(item) THEN
            this.items.REMOVE(item)
            calculateTotal()
        END IF
    END METHOD

    // Add ingredient specifically (helper method per diagram)
    PUBLIC METHOD addIngredientToShoppingList(): Void
        // Logic to convert a selected recipe ingredient into an Item
        selectedIngredient = GET_SELECTED_INGREDIENT()
        newItem = NEW Item(selectedIngredient.name, selectedIngredient.qty)
        addItem(newItem)
    END METHOD

    // Calculate the total price of the cart
    PUBLIC METHOD calculateTotal(): Float
        tempTotal = 0.0
        FOR EACH item IN this.items
            tempTotal = tempTotal + (item.price * item.Quantity)
        END FOR
        this.totalPrice = tempTotal
        RETURN this.totalPrice
    END METHOD

    PUBLIC METHOD clearList(): Void
        this.items.CLEAR()
        this.totalPrice = 0.0
    END METHOD
END CLASS
```

Item:

```
CLASS Item
   PRIVATE attributes:
      ItemId: String
      name: String
      Quantity: Integer
      price: Float

   PUBLIC METHOD addItemToCart(): Void
      ShoppingList.addItem(this)
   END METHOD

   PUBLIC METHOD viewItemDetails(): String
      RETURN "Name: " + this.name + ", Qty: " + this.Quantity + ", Price: $" + this.price
   END METHOD
END CLASS
```

Recipe:

```
CLASS Recipe
   PRIVATE attributes:
      recipeID: String
      Title: String
      ImageUrl: String
      description: String
      Ingredients: List<String>
      instructions: List<String>
      review: List<Review>
      like: Integer
      author: String
```

```
// Create a new recipe
PUBLIC METHOD createRecipe(): Recipe
    newRecipe = NEW Recipe()
    newRecipe.Title = GET_INPUT("Title")
    newRecipe.Ingredients = GET_INPUT("Ingredients")
    RETURN newRecipe
END METHOD

// Calculate Average Rating
PUBLIC METHOD calculateAverageRating(): Float
    totalRating = 0
    count = 0
    FOR EACH r IN this.review
        totalRating = totalRating + r.rating
        count = count + 1
    END FOR

    IF count > 0 THEN
        RETURN totalRating / count
    ELSE
        RETURN 0.0
    END IF
END METHOD

PUBLIC METHOD incrementLikes(): Void
    this.like = this.like + 1
    DATABASE.UPDATE_LIKES(this.recipeID, this.like)
END METHOD

PUBLIC METHOD addReview(): Void
    newReview = NEW Review()
    newReview.writeReview()
    this.review.ADD(newReview)
```

```
        calculateAverageRating()
    END METHOD
END CLASS
```

Review:

```
CLASS Review
    PRIVATE attributes:
        ReviewID: String
        rating: Integer
        comment: String
        author: String
        datePosted: Date

    PUBLIC METHOD writeReview(): Void
        this.rating = GET_INPUT("Star Rating") // 1-5
        this.comment = GET_INPUT("Comment")
        this.datePosted = CURRENT_DATE()
    END METHOD
END CLASS
```

For you Page:

```
CLASS ForYouPage
    PRIVATE attributes:
        preferences: List<String>
        savedRecipes: List<Recipe>
        likedRecipes: List<Recipe>
        trendingRecipes: List<Recipe>
        recipeList: List<Recipe>
        algorithmVersion: String

    // Main method to generate the feed
```

```
PUBLIC METHOD browseForYouPage(): List<Recipe>
    recommendations = generateRecommendations()
    trending = getTrendingRecipes()

    finalFeed = MERGE(recommendations, trending)
    RETURN finalFeed
END METHOD

// Filter recipes based on preferences
PUBLIC METHOD generateRecommendations(): List<Recipe>
    matches = NEW List<Recipe>()
    allRecipes = DATABASE.GET_ALL_RECIPES()

    FOR EACH recipe IN allRecipes
        IF matchPreferences(recipe) THEN
            matches.ADD(recipe)
        END IF
    END FOR
    RETURN matches
END METHOD

// Helper to check preferences
PUBLIC METHOD matchPreferences(recipe: Recipe): Boolean
    FOR EACH tag IN this.preferences
        IF recipe.description CONTAINS tag OR recipe.Ingredients CONTAINS tag THEN
            RETURN True
        END IF
    END FOR
    RETURN False
END METHOD

PUBLIC METHOD getTrendingRecipes(): List<Recipe>
    // Logic to find recipes with high likes in the last 24h
```

```
        RETURN DATABASE.QUERY("SELECT * FROM Recipes ORDER BY likes DESC
LIMIT 10")
    END METHOD

    PUBLIC METHOD searchRecipes(): List<Recipe>
        keyword = GET_USER_INPUT("Search Term")
        RETURN DATABASE.QUERY("SELECT * FROM Recipes WHERE Title LIKE '%" +
keyword + "%'")
    END METHOD
END CLASS
```