FINAL PROJECT REPORT

Rapid Application Development

Submitted by: Thi Uyen Mai

Project: Personal Recipe Tracker Application

1. INTRODUCTION

1.1 Project Background

The Personal Recipe Tracker is a Windows Forms desktop application developed as a course project for CSC 660 - Rapid Application Development. The application addresses the common challenge of managing personal recipes, planning meals, and organizing grocery shopping in an efficient and user-friendly manner.

In today's busy lifestyle, individuals struggle with meal planning, recipe organization, and grocery shopping coordination. Existing solutions are often overly complex or lack essential features. This project aims to provide a focused, practical solution that combines recipe management, meal planning, and grocery list generation into a single application.

1.2 Project Objectives

The primary objectives of this project were to:

1. Develop a fully functional Windows Forms application using C# and .NET 8.0
2. Implement a SQL Server database with proper normalization and relationships
3. Create an intuitive user interface with five main forms for different functionalities
4. Implement data access layer using Dapper ORM for efficient database operations
5. Utilize stored procedures for data integrity and security
6. Demonstrate understanding of rapid application development principles
7. Apply three-tier architecture pattern for separation of concerns

1.3 Technology Stack

The application was built using modern Microsoft technologies:

- **Programming Language:** C# (.NET 8.0)
- **User Interface Framework:** Windows Forms (WinForms)
- **Database:** Microsoft SQL Server
- **Object-Relational Mapping:** Dapper (Micro-ORM)
- **Database Provider:** System.Data.SqlClient

- **Development Environment:** Microsoft Visual Studio 2022

1.4 Project Scope

The application focuses on personal recipe management with the following core features:

- **Recipe Management:** Create, view, search, and delete recipes
- **Ingredient Management:** Pre-populated ingredient database with unit tracking
- **Meal Planning:** Weekly calendar-based meal planning interface
- **Grocery List Generation:** Automated grocery list creation from meal plans
- **Dashboard Statistics:** User statistics and quick navigation

The application is designed for single-user scenarios, with the foundation laid for potential multi-user expansion in future iterations.

2. IMPLEMENTATION

2.1 System Architecture

The application follows a three-tier architecture pattern, ensuring clear separation of concerns:

PRESENTATION LAYER:

The presentation layer consists of five Windows Forms:

- **PersonalDashboardForm:** Central hub displaying statistics and navigation
- **PersonalAddRecipeForm:** Interface for creating new recipes with ingredients
- **PersonalViewRecipesForm:** Recipe browsing, searching, and detail viewing
- **PersonalMealPlannerForm:** Weekly meal planning calendar interface
- **PersonalGroceryListForm:** Automated grocery list generation

BUSINESS LOGIC LAYER:

The business logic layer includes:

- **Model Classes:** Recipe, User, Ingredient, RecipeIngredient, MealPlan
- **Interface Definition:** IDataConnection contract for data operations
- **Business Rules:** Validation logic and data transformation

DATA ACCESS LAYER:

The data access layer implements:

- **SqlConnector:** Implements IDataConnection using Dapper ORM
- **GlobalConfig:** Manages application configuration and connection strings
- **Database Operations:** All operations use stored procedures

DATABASE LAYER:

SQL Server database with:

- Five normalized tables with proper relationships
- Seven stored procedures for all database operations
- Foreign key constraints ensuring data integrity

2.2 Database Design

The database schema consists of five main tables:

Users Table: Stores user account information with UserID as primary key.

Recipes Table: Contains recipe details including name, description, prep time, cook time, instructions, and user association. Foreign key relationship to Users table.

Ingredients Table: Pre-populated list of ingredients with name and unit of measurement. Includes common US and Vietnamese ingredients.

RecipeIngredients Table: Junction table implementing many-to-many relationship between Recipes and Ingredients. Contains RecipeID, IngredientID, and Quantity with composite primary key.

MealPlans Table: Stores meal planning information linking users, recipes, dates, and meal types (Breakfast, Lunch, Dinner). Foreign key relationships to both Users and Recipes tables.

All relationships are properly normalized with foreign key constraints ensuring referential integrity. Cascade delete is implemented for RecipeIngredients when recipes are deleted.

2.3 Data Access Implementation

The data access layer uses Dapper ORM for efficient database operations:

STORED PROCEDURES:

All database operations use stored procedures for security, performance, and maintainability:

- **sp_Recipes_Insert:** Creates new recipes with OUTPUT parameter for RecipeID

- **sp_RecipeIngredients_Insert:** Links ingredients to recipes
- **sp_MealPlans_Insert:** Creates meal plan entries
- **sp_GetRecipesByUser:** Retrieves user's recipes with ingredients
- **sp_GetUserMealPlansForWeek:** Retrieves weekly meal plans
- **sp_GetGroceryListForWeek:** Generates aggregated grocery list
- **sp_Recipes_Update:** Updates existing recipes

DAPPER IMPLEMENTATION:

The SqlConnector class implements IDataConnection interface using Dapper:

- DynamicParameters for stored procedure calls
- Automatic connection management with 'using' statements
- Manual mapping for complex relationships (RecipeIngredients with nested Ingredient objects)
- Comprehensive error handling for missing stored procedures

2.4 User Interface Implementation

Each form was designed and implemented with specific functionality:

PERSONAL DASHBOARD FORM:

- Displays total recipe count and this week's meal count
- Provides navigation buttons to all features
- Refreshes statistics after returning from child forms

PERSONAL ADD RECIPE FORM:

- Dynamic ingredient unit display (updates when ingredient is selected)
- Input validation for required fields
- Ingredient list management with add/remove functionality
- Saves recipe and all ingredients to database

PERSONAL VIEW RECIPES FORM:

- Real-time search functionality filtering recipes by name
- Detailed recipe view showing all information and ingredients
- Recipe deletion with confirmation dialog
- Formatted display of recipe details

PERSONAL MEAL PLANNER FORM:

- Weekly calendar view (Monday through Sunday)
- Three meal types per day (Breakfast, Lunch, Dinner)
- Visual indication of planned meals
- Week navigation (Previous/Next buttons)
- Recipe assignment to meal slots

PERSONAL GROCERY LIST FORM:

- Week-based grocery list generation
- Automatic ingredient aggregation from meal plans
- Quantity summation for ingredients used in multiple recipes
- Printable grocery list format

3. PROBLEM SOLVING

3.1 Challenge 1: Week Calculation Logic

PROBLEM:

Need to calculate week start (Monday) and end dates for meal planning queries, accounting for different starting days of the week. The calculation needed to handle edge cases like Sunday (which is day 0 in C# DayOfWeek enum).

SOLUTION:

Created helper method GetWeekStart() that calculates the Monday of the week containing a given date. Special handling for Sunday (day 0) by subtracting 6 days instead of the standard calculation.

IMPLEMENTATION:

```
private DateTime GetWeekStart(DateTime date)
{
   var dayOfWeek = (int)date.DayOfWeek;
   var daysToSubtract = dayOfWeek == 0 ? 6 : dayOfWeek - 1;
   var monday = date.AddDays(-daysToSubtract);
   return monday.Date;
}
```

RESULT:

Consistent week boundaries across meal planning and grocery list features, with proper handling of all days of the week.

3.2 Challenge 2: Form State Management

PROBLEM:

Dashboard statistics needed to refresh after users returned from child forms (e.g., after adding a recipe, the total recipe count should update).

SOLUTION:

Implemented LoadDashboardData() refresh in navigation button click handlers. After child forms close (ShowDialog returns), the dashboard refreshes its statistics.

IMPLEMENTATION:

```
private void btnAddRecipe_Click(object sender, EventArgs e)
{
   var addRecipeForm = new PersonalAddRecipeForm(_currentUser);
   addRecipeForm.ShowDialog();
   LoadDashboardData(); // Refresh statistics after form closes
}
```

RESULT:

Dashboard always displays current, up-to-date statistics after any operation.

4. CONCLUSION

4.1 Project Summary

The Personal Recipe Tracker application successfully demonstrates proficiency in rapid application development using C# and SQL Server. The implementation follows best practices including three-tier architecture, separation of concerns, use of stored procedures for data access, and comprehensive error handling.

The application is fully functional with all core features implemented:

- Recipe creation, viewing, searching, and deletion
- Ingredient management with dynamic unit display
- Weekly meal planning with visual calendar interface
- Automated grocery list generation with ingredient aggregation
- Dashboard statistics with real-time updates

4.2 Key Achievements

The project successfully achieved all primary objectives:

1. **Complete Application:** All five forms are fully functional and integrated

2. **Robust Database Design:** Well-normalized database schema with proper relationships and constraints
3. **Efficient Data Access:** Dapper ORM implementation with stored procedures for optimal performance
4. **User-Friendly Interface:** Intuitive forms with validation and error handling
5. **Feature Completeness:** All planned features implemented and tested

4.3 Learning Outcomes

This project provided valuable experience in:

- Windows Forms application development and event-driven programming
- SQL Server database design, normalization, and relationship management
- Dapper ORM usage, including complex object mapping and stored procedure integration
- Stored procedure design and execution for security and performance
- Three-tier architecture implementation and separation of concerns
- Error handling, input validation, and user experience design
- Rapid application development principles and iterative development

4.4 Future Enhancements

Potential improvements for future iterations include:

- Multi-user support with authentication and user management
- Recipe categories, tags, and advanced search
- Nutrition information tracking and calorie calculation
- Recipe images support for visual recipe display
- Export functionality (PDF, Excel) for recipes and grocery lists
- Recipe sharing between users