RECIPEBOX BLUEPRINTS

MVP 1.0

Table of Contents

1 Introduction

2 Agile w/ Blueprints' Framework

3 Agile w/ Blueprints Product

3.1 System Blueprints

3.1.1 [Business] Domain Model

3.1.1.1 Domain Use Cases

3.1.1.1.1 Feed the family for a week

3.1.1.2 Domain Class Diagram

3.1.2 Product Vision

3.1.2.1 Problem Statement

3.1.2.2 Product Business Objectives

3.1.2.3 Product Roadmap

3.1.2.3.1 MVP 1.0 Project

3.1.2.3.1.1 Release Features/Functions

3.1.2.3.1.2 Return On Investment (ROI)

3.1.2.3.2 MVP 1.1 Project

3.1.2.3.2.1 Release Features/Functions

3.1.2.3.2.2 Return On Investment (ROI)

3.1.3 System Spec

3.1.3.1 Solution Model

3.1.3.1.1 As-Is Solution

3.1.3.1.1.1 Issues with the As-Is Solution

3.1.3.1.2 To-Be Solution

3.1.3.1.2.1 To-Be Solution Use Cases

3.1.3.1.2.2 To-Be Solution Class Diagram

3.1.3.2 UI Design

3.1.3.2.1 System User Personas

3.1.3.2.2 UI Conceptual Framework & UI Standards

3.1.3.2.3 UI Navigation Map

3.1.3.2.4 Bodies of Business Objects

3.1.3.3 Detailed Spec
3.1.3.3.1 Functional Requirements
3.1.3.3.1.1 Business Rule Rqts
3.1.3.3.1.2 UI Forms - Detailed Rqts
3.1.3.3.1.3 Business Automation Rqts
3.1.3.3.1.4 Import / Export / Print Rqts
3.1.3.3.1.5 Reports
3.1.3.3.1.6 Security Rqts
3.1.3.3.1.7 External System Interfaces
3.1.3.3.1.8 System Administration / Operations Rqts
3.1.3.3.1.9 Migration Rqts
3.1.3.3.2 Non-Functional Requirements
3.1.3.3.2.1 Design Constraints / Deployment / Installation / Configuration Rqts
3.1.3.3.2.2 Performance / Load / Concurrency Rqts
3.1.3.3.2.3 Scalability / Capacity Rqts
3.1.3.3.2.4 Robustness / Availability / Service-ability / Fault Tolerance Rqts
3.1.3.3.2.5 Internationalization Rqts
3.1.3.3.2.6 Product Lifespan Rqts
3.1.3.3.3 Build the Detailed Spec
3.1.4 System Design
3.1.4.1 Functional Design
3.1.4.1.1 Model-View Design
3.1.4.1.2 Implementation Model Design
3.1.4.1.3 Business Object Design
3.1.4.1.4 Versioning the Implementation Model
3.1.4.1.5 Scalable, Serverless Design
3.1.4.1.6 UI Framework Class Design
3.1.4.1.7 Simple Search and Edit RecipeCard Use Case
3.1.4.1.8 Window Navigation Design
3.1.4.1.9 Edit (CRUD) a Body of Business Objects Design
3.1.4.1.10 Maintain Proper View State - Design
3.1.4.1.11 Enforcing Business Rules - Design
3.1.4.1.12 SeachRecipeBox Design
3.1.4.1.13 Import AllRecipes.com Design
3.1.4.1.14 Email GroceryList Design

3.1.4.2 Security Design
3.1.4.3 Concurrency Design
3.1.4.4 DataStore Design
3.1.4.4.1 eRecipeBox Data Dictionary
3.1.4.5 Component Design
3.1.4.5.1 eRecipeBox Components
3.1.4.5.2 SDKs & Third Party Components
3.1.4.6 Topology Design
3.1.4.6.1 Service Design
3.1.4.6.2 Network Design
3.1.5 System Implementation
3.1.5.1 Coding Standards
3.1.5.2 Source Code
3.1.5.3 Version Control
3.1.5.4 Build the System
3.1.5.5 System Implementation Metrics
3.2 Test Blueprints
3.2.1 Test Strategy
3.2.1.1 System Test Approach
3.2.1.2 System Test Suites
3.2.1.3 System Test Requirements Levied on the System
3.2.2 Test Suite Design
3.2.3 Test Suite Implementation
3.2.3.1 System Test Data
3.2.3.1.1 Large System Test Data
3.2.4 Automated Test Design
3.2.4.1 Automated System Test Functional Design
3.2.4.2 Automated System Test Component Design
3.2.4.2.1 Automated System Test Components
3.2.4.2.2 SDKs & 3rd Party Components
3.2.4.3 Topology Design
3.2.5 Automated Tests Implementation
3.2.5.1 Automated Test Coding Standards
3.2.5.2 Automated Test Source Code
3.2.5.3 Automated Test Version Control

3.2.5.4 Build System Tests
3.2.5.5 Automated Test Executions
3.2.5.5.1 MVP 1.0 Test Execution
3.2.6 Requirement Traceability Matrix (RTM)
3.3 User Education Blueprints
3.3.1 User Education Strategy
3.3.2 User Education Design
3.3.3 User Education Implementation
4 Agile w/ Blueprints Team
4.1 AwB Project Sub-Teams
5 Agile w/ Blueprints Process
5.1 Development Process Description
5.2 Execute the Process
5.2.1 Risk Assessment
5.2.2 eRecipeBox MVP 1.0 - Sprint Plan
5.3 Environments
5.3.1 Project Leadership Environment
5.3.2 Dev Team Environment
5.3.2.1 Configure 3rd Party Services
5.3.2.2 Multi-tier Dev Environments
5.3.2.2.1 Install SQL Server on another LAN PC
5.3.2.2.2 Install RecipeBoxDataService on Dev PC
5.3.2.2.3 Deploy RecipeBoxDataService on another PC
5.3.3 Test Team Environment
5.3.4 UE Team Environment
5.3.5 Production Environment
5.3.5.1 Install eRecipeBox
5.3.5.2 Configure eRecipeBox
5.3.5.2.1 Local, single-user configuration
5.3.5.2.2 Two-tier configuration
5.3.5.2.3 Three-tier configuration
6 Hands on Exercises
6.1 Functional Enhancements
6.2 Deployment Backlog Tasks
6.3 Design Backlog Tasks

6.4 Security Backlog Tasks
6.5 Test Backlog Tasks

1 Introduction

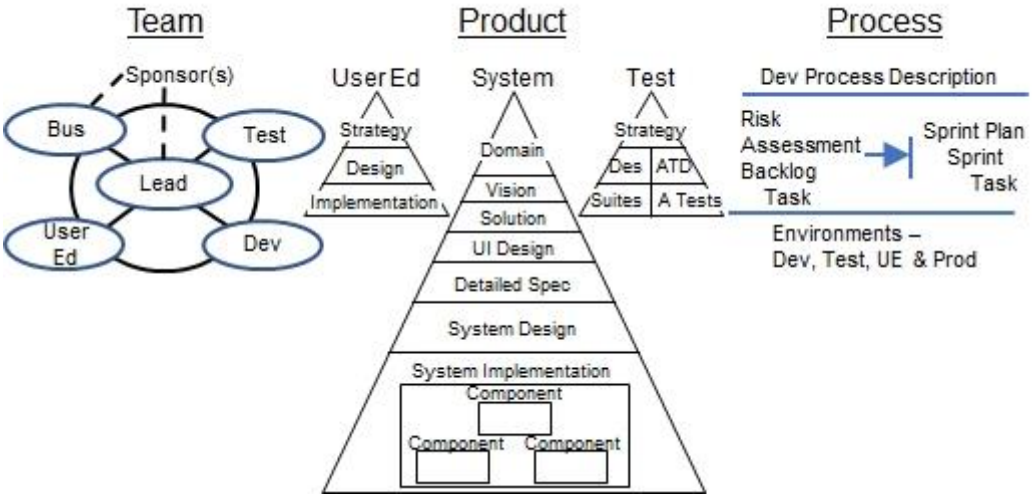
Agile w/ Blueprints (AwB) is a lightweight yet comprehensive process for building production-quality business applications. eRecipeBox is an instructional sample application built to illustrate AwB in action. This sample application complements the book- *How I Built a Business Application using Agile w/ Blueprints,* available on Amazon for $9.99.

2 Agile w/ Blueprints' Framework

AwB's framework is formed by applying proven engineering practices, leveraging software's unique development opportunities, and applying guidelines defined in

- Agile and Scrum
- Rational Unified Process (RUP)
- C4 Model, and
- Domain-Driven Design (DDD)

…to developing business apps. Object modeling, levels of detail, and rapid iterations drive AwB's structural framework of Team, Product and Process.



Agile w/ Blueprints Framework

A Team follows a Process to create a Product. The Product comprises three parts – User Education, System, and Test. Each of these is modeled in multiple levels of detail. For example, the System levels of detail are:

1. Object Model of the Business Domain
2. Product Vision (including Roadmap)
3. Object Model of the Solution
4. User Interface Design
5. Detailed Specification
6. System Design
7. System Implementation

AwB prescribes a specific process to model business processes in the context of the solution and effectively communicate them to the entire project team. Analysts, architects, and developers rapidly iterate with Business Representatives to discover, create, refine, and evolve the System's functionality, design, and implementation. AwB emphasizes designing for the long term, so "Stable Design" becomes a project milestone instead of a separate project phase.

Decisions are delegated to various team members. Each team member is accountable for their contributions and owns their part of the blueprints and implementation. Concurrently, the Test and User Education teams iterate on their strategies, designs, and implementations. A Risk Assessment drives the Sprint planning process. Early Sprints emphasize building end-to-end functionality and reducing high-exposure risks (which includes stabilizing the design). Later Sprints address low-risk details, System polish & punch-list tasks.

The remaining sections of these materials illustrate the Agile w/ Blueprints process through a working instructional example: the eRecipeBox app. If you haven't done so, watch this video to see the final product in action.

The following sections present - eRecipeBox's Product artifacts, then its Team structure, and conclude with Process steps to build the Product.

Here, you can download:

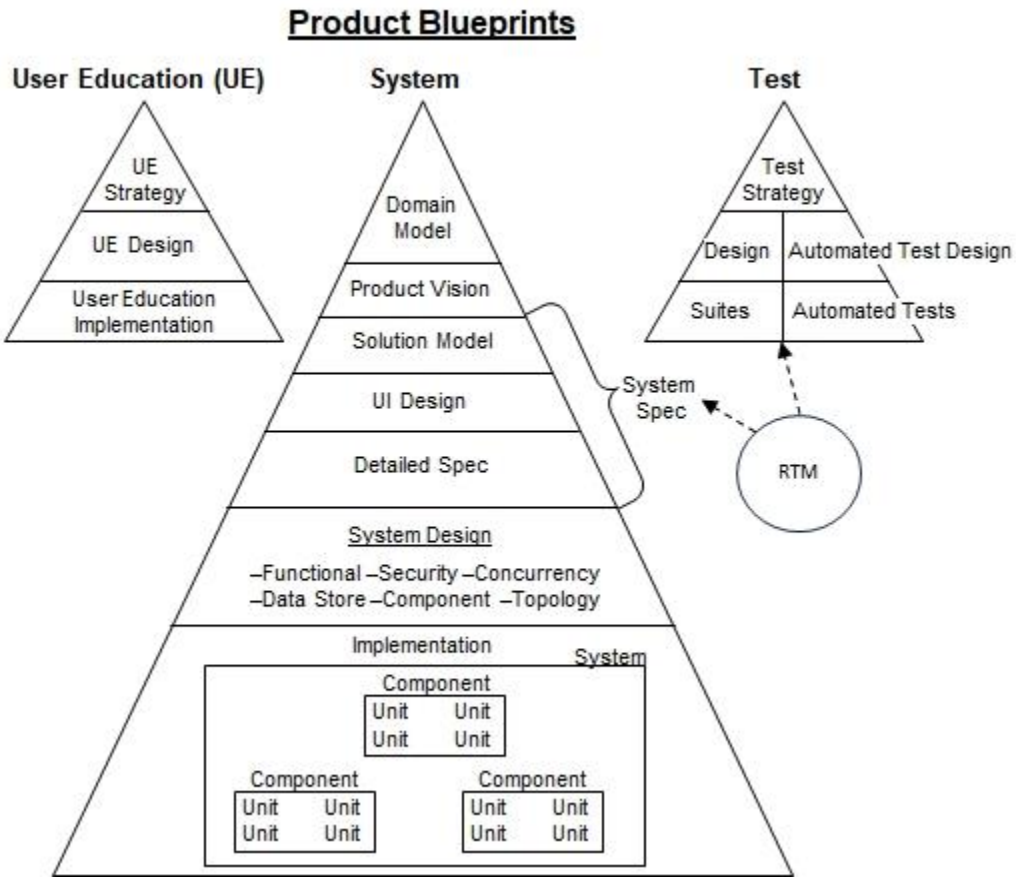
A single user version of eRecipeBox.

The eRecipeBox Source Package which also contains these materials.

3 Agile w/ Blueprints Product

AwB's **Product** comprises three views – the **System** itself, end **User Education,** and **Test**. Each of these parts requires high-level decisions (strategy) that drive their design and implementation.

The following diagram summarizes the 16 artifacts that comprise AwB's **Product**. These decisions are *not* made sequentially in a top-down fashion but iteratively & evolutionary, as discussed in Section 4.3. However, as team members make their decisions, they file each decision in its appropriate artifact so other teammates can easily find them.



System – The system is modeled in several levels of detail.

[Business] Domain Model – An object model of the business context, independent of any particular manual or automated workflow processes for conducting the business. *Level of detail context*: Each company that participates in the industry represents a black box.

Product Vision – Defines the scope of the business problem to be solved and quantifies the business and technical objectives for the product. *Level of detail context*: Lists the goals, objectives, and benefits the product will deliver to the organization when deployed. Defines product "success" for the business and the organization. Justifies the financial investment for developing the solution.

System Specification – Specifies all decisions that impact external actors, modeled in three levels of detail - Solution Model, UI Design and Detailed Spec. *Level of detail context*: Each System is a black box. End user actors interact with the system(s).

Solution Model – Contains two models of the business solution highlighting the functionality of the System.

As-Is Solution – An object model of the current solution in production. Typically illustrates the workflow using legacy systems and manual tasks.

To-Be Solution – An object model of the proposed solution. *Level of detail context*: The System is the black box. It models how the System interacts with other systems and external actors. For solutions with a wide reach, it models each corporate department and their interactions first, then, when appropriate, zooms in and models end users interacting with systems as a black box.

UI Design – Communicates the end user experience paradigm and window navigation map. *Level of detail context*: Identifies UI platforms, UI standards, key UI concepts, and user experience paradigm. Identifies the UI forms in the System, e.g., the site map for web applications.

Detailed Specification – Specifies the lowest level details visible to system users. *Level of detail context*: UI forms/web pages specifics, data types and formats, business rules, external APIs, report layouts, etc. *Level of detail context*: External actors notice these decisions. Using a home project analogy, wall paint color selections are documented in the Detailed Spec.

System Design – Captures design decisions made *inside* the System black box in six views. *Level of detail context*: External actors **do not** notice system design decisions.

Functional Design – An object model of the major classes that implement the system functionality, design patterns and design conventions. Describes the part of the design that implements the core functionality, ignoring decisions required in the other views (concurrency, security, etc.).

Security Design – Describes the design for user authentication, authorization, how the system protects secrets, and auditing access to the system resources.

Concurrency Design – Describes concurrent agents (both internal and external) and how the system retains data integrity & delivers acceptable performance with this concurrency.

DataStore Design – Describes the logical and physical structure for all persisted data. For example, it contains Entity/Relationship diagrams for relational stores, DOMs or Key-Values for NoSQL stores, networks for graph databases, format layout specifications for flat files, etc.

Component Design – Presents the System's software components (purchased and developed), their responsibilities, and their dependencies.

Topology Design – Presents the logical and physical deployment structure of the system. The Service Design presents the services and their communication paths. The Network Design specifies the physical network nodes, connections, firewalls, etc. It also includes which main program System Components are deployed to which nodes.

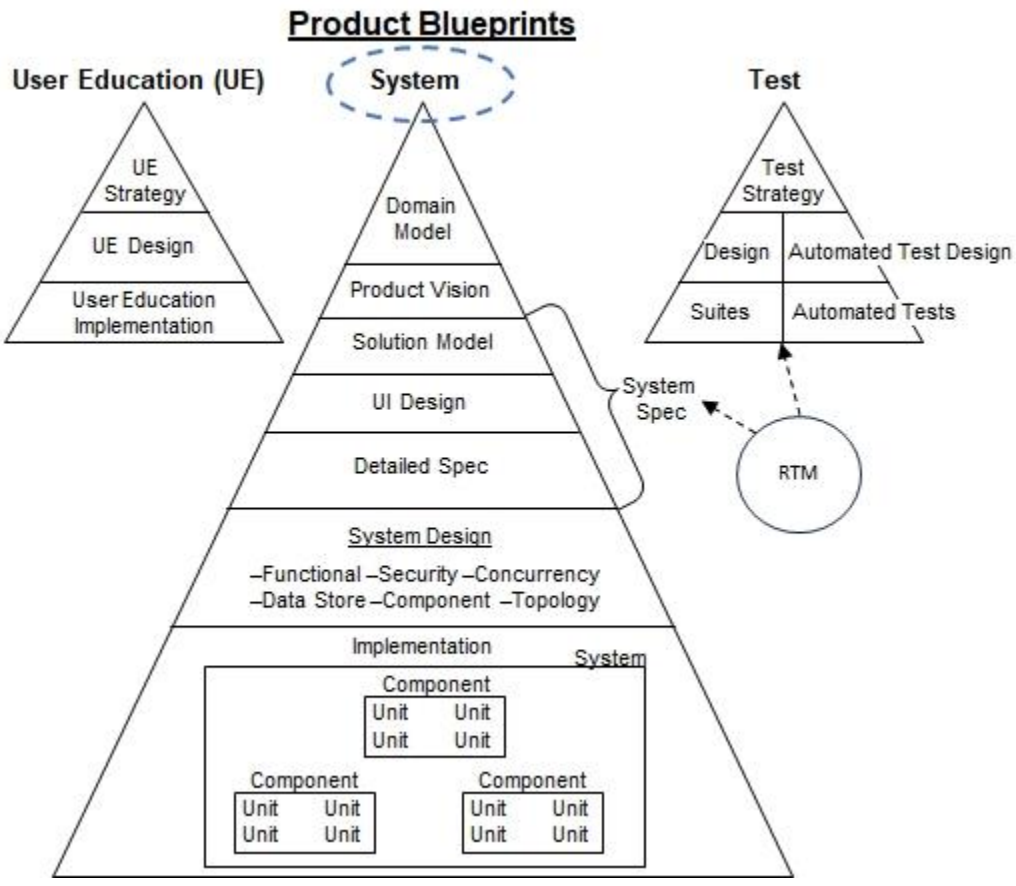
System Implementation – The System Implementation comprises all source code, build scripts, DDL, global reference data, application installation scripts, etc. assembled to form the deployed components.

User Education (UE) – User Education develops skills and assists end users to use the System effectively. UE decisions are modeled in three levels of detail, Strategy, Design, and Implementation.

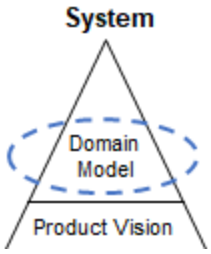
Test - Verifies the System performs according to the Product Vision and System Spec. System Test also measures the quality of the System and User Education. Test decisions are modeled in three levels of detail, Strategy, Design, and Implementation. Test often includes automated test scripts to support AwB's rapid iterations.

3.1 System Blueprints

AwB System artifacts model System decisions in seven levels of detail, Domain Model, Product Vision, Solution Model, UI Design, Detailed Spec, System Design, and System Implementation.



3.1.1 [Business] Domain Model



The [Business] Domain models the business environment, independent of any solution. It provides valuable business context to the project team. Following is eRecipeBox's Business Domain Model.

3.1.1.1 Domain Use Cases

Actors: Meal Scheduler, Grocery shopper, Cook, Family members

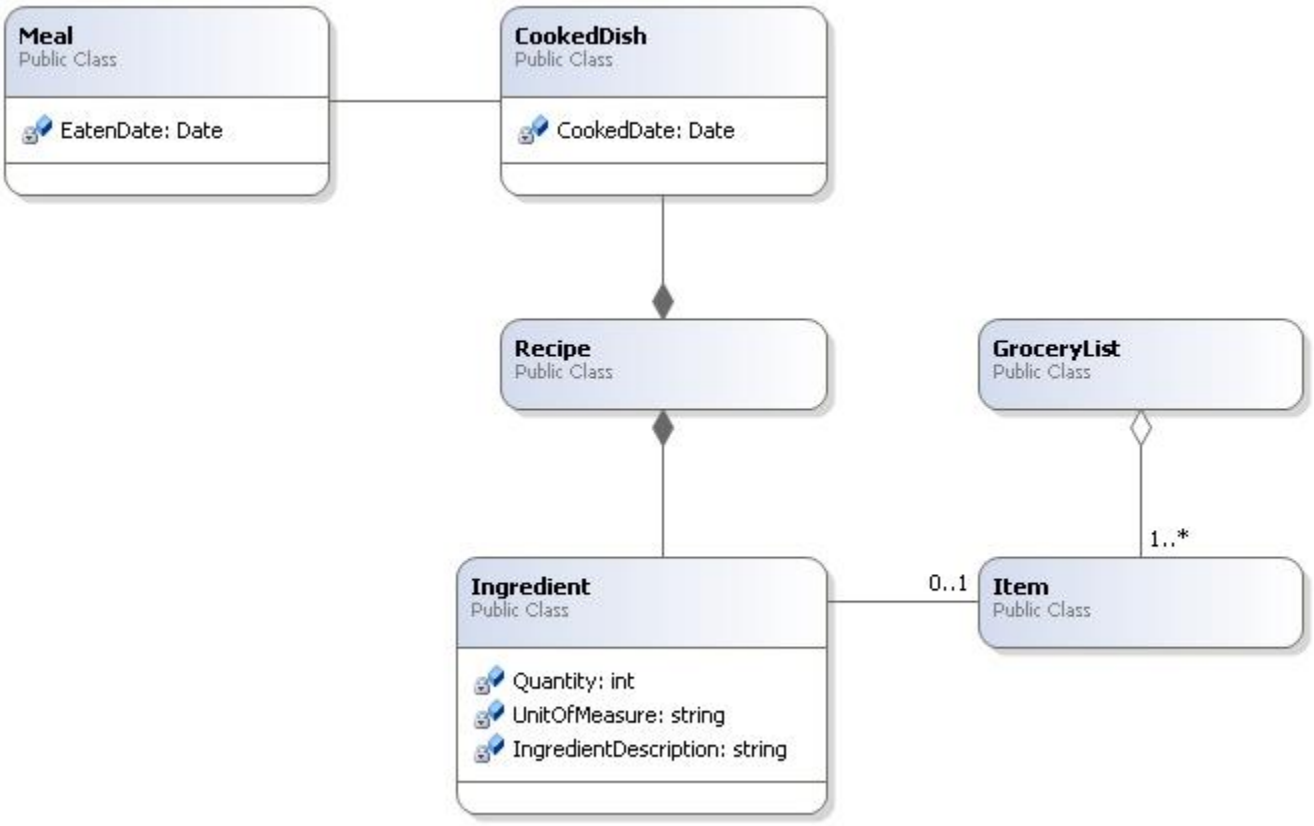
3.1.1.1.1 Feed the family for a week

1. Meal Scheduler reviews the upcoming family schedule for the week and determines which days meals need to be prepared.
2. Incorporating various inputs such as family member preferences, grocery inventory, grocery budget, leftovers and dish recency, Meal Scheduler **schedules** one or more recipes for each day of the following week.
3. Meal Scheduler **creates a grocery list** for the items required for the planned recipes, less items already in the refrigerator and pantry.
4. Grocery shopper **purchases the items** on the grocery list from the grocery store.
5. Each day, cook **prepares the dishes** for the meal scheduled for that day.
6. Family members **eat** the cooked meal.

Notes and Variations:

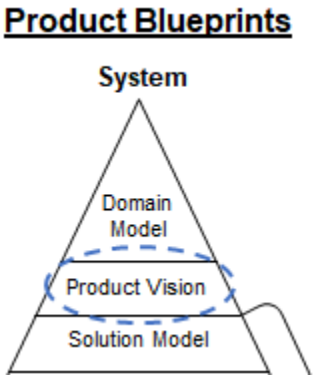
- Meal Scheduler typically schedules just the dinner meal for each day. Family members are on their own to prepare their breakfasts and lunches.
- Meal Scheduler always includes "staples" when creating the grocery list. Staples are defined as items the family likes to always have on hand – drinks, milk, fruit, peanut butter, snacks, eggs, breakfast cereal, etc.
- Any family member may request item(s) to be added to the grocery list.
- Grocery shopper may need to shop at multiple grocery stores to purchase all items on the grocery list.
- Leftovers are either frozen or packaged & placed in the refrigerator.
- Leftovers are taken to work/school for lunch or served as part of another dinner.

3.1.1.2 Domain Class Diagram



A **Recipe** is a list of **Ingredients** and instructions to combine and cook the ingredients in order to create a **CookedDish**. Each Ingredient lists a food **Item**, quantity and preparation instructions. A **GroceryList** is a collection of items and their quantity that are needed to purchase from a grocery store. The family eats a **Meal**, which is a collection of **CookedDishes**.

3.1.2 Product Vision



The Product Vision defines the scope and challenges in the business domain that the Product will address and then presents the Product's main objectives for addressing these challenges. All efforts invested in developing the Product, direct or indirect, support the Product Vision. The last section, Product Roadmap, lays out the sequence of functionality the Product will deliver through a series of Product Releases.

3.1.2.1 Problem Statement

Dad works at home while Mom works long hours in the office each week. As a result, Dad is responsible for meal planning, grocery shopping, and cooking. Since Dad also works, he doesn't devote much time to these activities. Specifically,

- Dad doesn't do any meal planning.
- As a result, around 4:30 PM, Dad scrambles to decide what to make for dinner based on what's currently in the refrigerator & pantry.
- Due to this lack of planning, Dad often has to make several, last-minute trips to the store to get missing ingredient(s).
- Dad frequently "punts" and makes grilled cheese or tuna sandwiches.

Cost summary: Wasted time (1-3 extra trips to the store each week). The family doesn't eat a variety of unhealthy dishes. The grocery list is created manually and often incomplete. When Dad does schedule meals, it's time-consuming and stressful.

3.1.2.2 Product Business Objectives

Health & variety:

- Eat a wide variety of dishes that are healthy
- Rotate & repeat favorite meals

Efficiency:

- Schedule meals for an upcoming week in 2-5 minutes. And quickly generate a grocery list from them.
- One trip to the grocery store per week.
- Adding new recipes from various sources – websites, cookbooks, friends & family - must be easy, 0-2 minutes.
- Must be easy to search & categorize recipes. Search by ingredient, rating, and/or category.

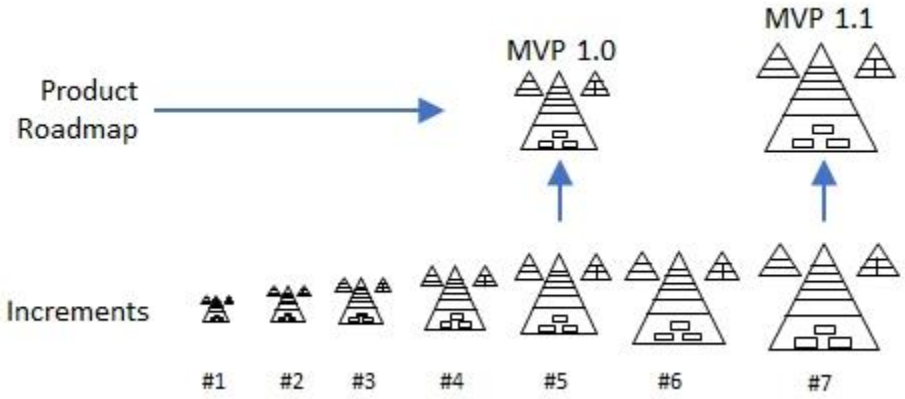
MVP 1.0 supports a single family's recipe box. eRecipeBox's long-term product vision is to be adopted mainstream and add community & socialization to meal planning. Current recipe products and websites support posting recipes and allowing other users to rate and comment the recipes. Searching by ingredient or dish type and rating is helpful, however, there are still too many search results making it difficult to find recipes that *my* family will enjoy. eRecipeBox adds community by supporting these use cases.

1. Each family creates a profile - demographics, dietary preferences, health, lifestyle, and "attitude toward cooking". Examples -
   - Two retired empty nesters, pescatarian (eat fish once per week), avoid sweets & fried food, active, like Mediterranean, Indian & Mexican, prefer healthy, simple & easy recipes. Not foodies.
   - Family of four, two teenagers (boy and girl), carnivores, like casseroles, burgers, pasta, chicken, pork, dairy & desserts, like Italian, American, Mexican & BBQ, daughter has nut allergy, kids are active, parents are not, prefer fast recipes with inexpensive ingredients because they have a busy schedule and a tight budget.
2. Families maintain their own variations of recipes on their RecipeCards in their eRecipeBox. Also can add their own notes.
3. Families rate the RecipeCards in their eRecipeBox (1-10)
4. Families opt to make their Profile and eRecipeBox visible for others to find (yet remain anonymous)
5. Other eRecipeBox users can search for profiles similar to theirs. This enables users to try the 9 and 10 rated RecipeCards of families like mine. This should significantly increase the probability *my* family will like the recipe.
6. Users can create groups. Members of the group share their eRecipeBoxes. Examples –

- Extended family members ("I want grandma's pound cake recipe")
- Social group (e.g., pickleball league members)
- Neighbors
- Families of all my son's classmates
- High school classmates
- Basically, any Facebook group is a potential eRecipeBox group

3.1.2.3 Product Roadmap

eRecipeBox product functionality is deployed through a series of Releases. Release 1.0, called the Minimal Viable Product (MVP), provides minimal features that benefit the family over their current As-Is Solution. Similarly, each Release is created by evolving a series of increments. Section 5 describes the AwB process for developing the Increments through a series of Sprints.



eRecipeBox's vision is to deploy MVP 1.0 into production and monitor six months of user experience before investing in MVP 1.1 development. Following is a summary of eRecipeBox's product roadmap.

MVP 1.0	MVP 1.1	V2.0	V3.0
-Desktop, laptop, touch tablet PC support -UserProfile authentication -One RecipeBox per UserProfile -Import RecipeCard from text, recipes website(s), GPT -Edit my RecipeCard including my rating -Add Keywords to RecipeCard -Search RecipeBox -Schedule meals -Generate GroceryList from schedule -Email GroceryList to phone -View RecipeCard to cook -Admin: Backup/Restore	-Mobile app for GroceryList -Print RecipeCard to .pdf -Import from additional website(s) -Reports on meal variety, healthy meals, favorites -Move GMail, Azure and GPT interfaces to server	-Mobile app for RecipeBox -eRecipeBox app updates itself automatically -Share my RecipeBox with other UserProfiles -Import RecipeCards from other user RecipeBoxes -Simple automated meal scheduling suggestions -Audit log of UserProfiles who edit RCs	-Enhanced UserProfile characteristics so users can find cooks like me -More sophisticated automated meal scheduling

3.1.2.3.1 MVP 1.0 Project

3.1.2.3.1.1 Release Features/Functions

MVP 1.0 scope is summarized in the Roadmap above.

3.1.2.3.1.2 Return On Investment (ROI)

The family plans to use eRecipeBox for decades, including extended family members so they can share RecipeCards. So the family is willing to invest $1000 in tools and 2 resources for 4 weeks to develop MVP 1.0.

3.1.2.3.2 MVP 1.1 Project

3.1.2.3.2.1 Release Features/Functions

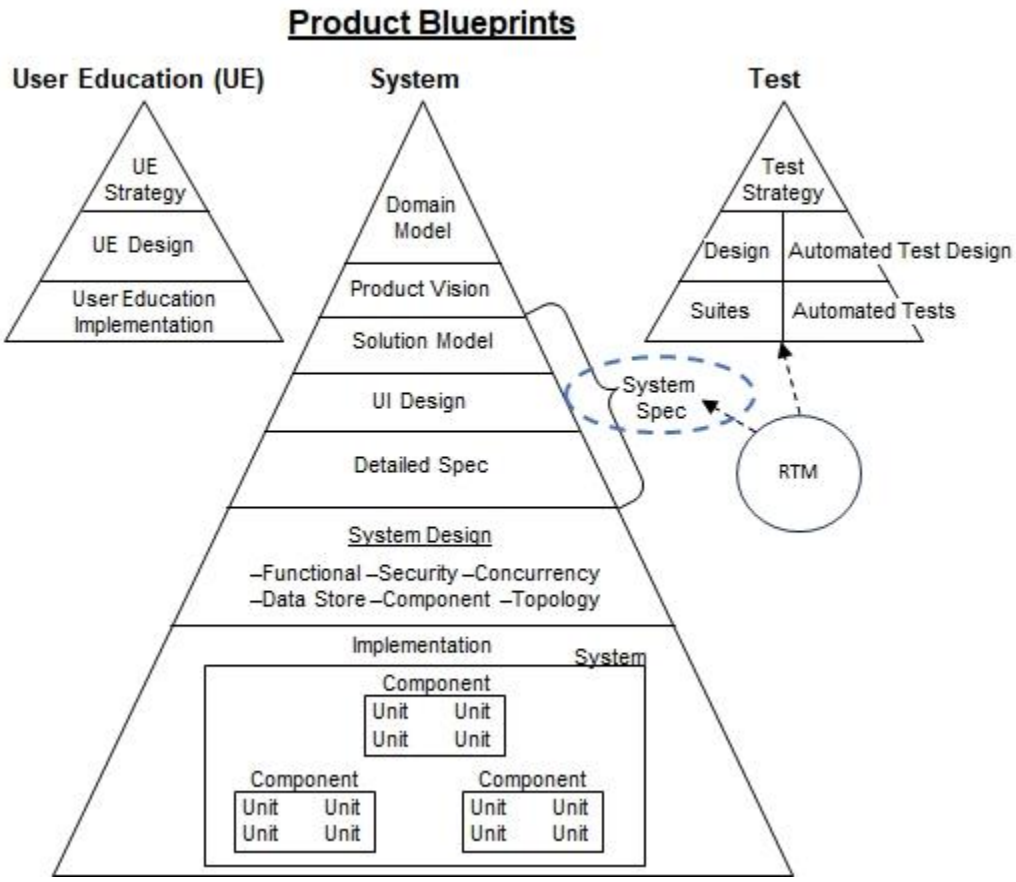
See Roadmap.

3.1.2.3.2.2 Return On Investment (ROI)

[To Be Completed before starting MVP 1.1 development.]

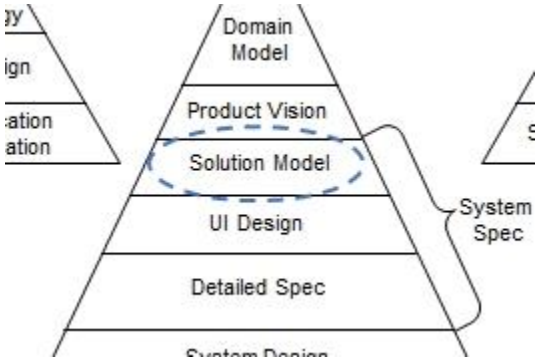
3.1.3 System Spec

The System Spec documents all external behavior of the System, behavior that impacts external system actors. It's organized in three levels of detail, Solution Model, UI Design and Detailed Spec.



3.1.3.1 Solution Model

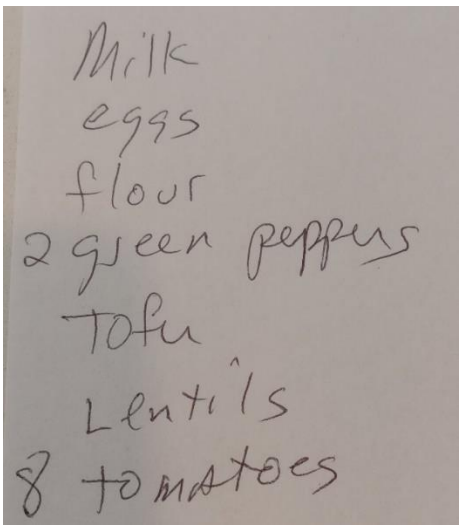
The Solution Model models the current As-Is Solution and narrates the high-level business workflows for the To-Be Solution (where the System serves as the central actor).



3.1.3.1.1 As-Is Solution

If a homeowner is getting a house renovation, the architect first measures the existing structure and reverse engineers its blueprints. Similarly, it's beneficial to model the current business processes to understand the current workflows and use it as a starting point for crafting the To-Be model. The As-Is model is often used as the starting point to ensure the business can migrate from the As-Is to the To-Be when deploying MVP 1.0.

The following pictures summarize the family's As-Is Solution… no narration is required.

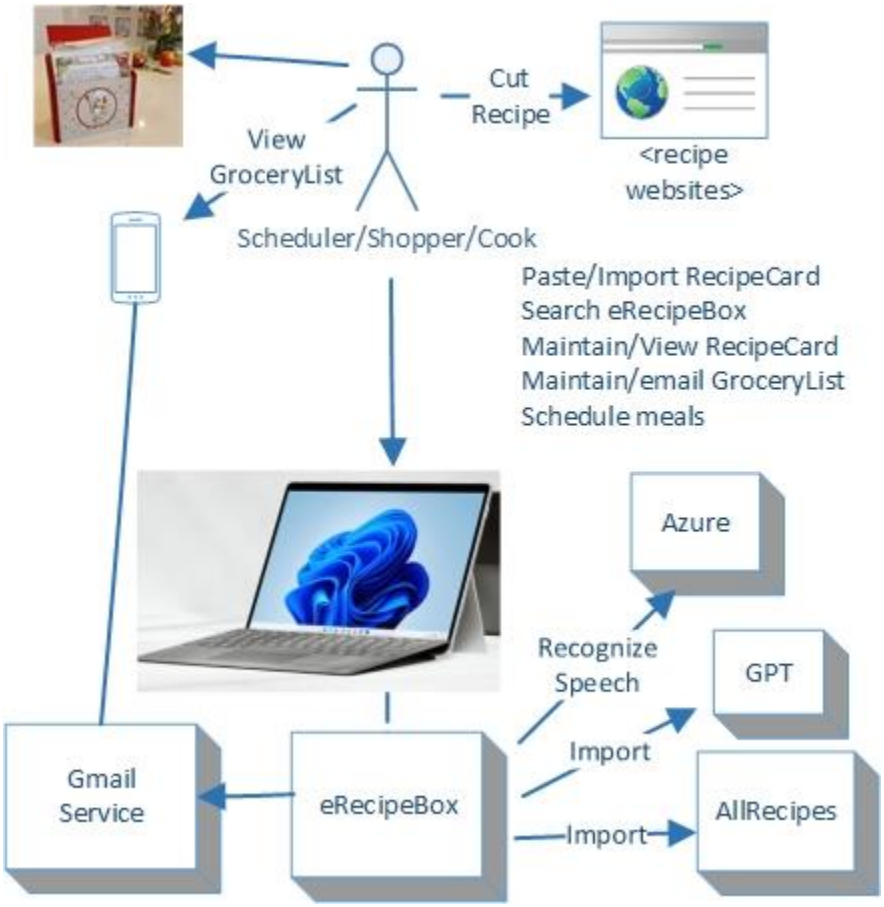


3.1.3.1.1.1 Issues with the As-Is Solution

See Section 6.1.2.1 in the Product Vision.

3.1.3.1.2 To-Be Solution

eRecipeBox stores the family's recipe cards digitally. Users use eRecipeBox to schedule meals, generate a GroceryList, and view RecipeCards to cook the meals. RecipeCards can be imported from text, AllRecipes.com website, or GPT.



eRecipeBox must run on the family's existing Windows PC computers.

eRecipeBox must be secure, but avoid requiring the user to enter a password when launching the app.

3.1.3.1.2.1 To-Be Solution Use Cases

Use cases are grouped into – Business Use Cases and Admin Use Cases.

3.1.3.1.2.1.1 Business Use Cases

Solution Model Actors: Meal Scheduler, Shopper, Cook, Family members

3.1.3.1.2.1.1.1 Maintain RecipeCard

1. Meal Scheduler searches for a RecipeCard using any term mentioned in title, description, source, ingredients, instructions or keywords.
2. Meal Scheduler selects and edits any information on the RecipeCard including ingredients, instructions, notes, keywords, cook time, ratings, etc. and saves it.

Variations:

- Meal Scheduler deletes RecipeCard from the eRecipeBox.
- Meal Scheduler imports a recipe found on a public website into the eRecipeBox.
- Meal Scheduler imports a recipe from AllRecipes.com website or GPT.
- Meal Scheduler imports a recipe from the family's existing recipe box, cook book, article, friend, etc.
- Meal Scheduler creates a new RecipeCard directly in the eRecipeBox app.

3.1.3.1.2.1.1.2 Schedule meals for the upcoming week

1. Meal Scheduler reviews recent meals.
2. Taking into account family member preferences (stored as MyRating on the RecipeCard), grocery inventory, grocery budget, leftovers and meal recency, Meal Scheduler searches eRecipeBox for candidate RecipeCards.
3. Meal Scheduler reviews the candidates and schedules RecipeCards to cover the meals for the upcoming week.
4. Meal Scheduler selects upcoming RecipeCards to prepare and adds their ingredients to the GroceryList, less items already in the home.
5. Meal Scheduler emails the GroceryList to the Shopper's email address.

Variations:

- Any family member can add, update, delete items from the GroceryList.

3.1.3.1.2.1.1.3 Shop for groceries

1. Shopper travels to grocery store.
2. Shopper views the emailed GroceryList on their mobile phone.
3. Shopper purchases items on the GroceryList.
4. Shopper stocks the refrigerator and pantry with the purchased items.
5. Shopper uses eRecipeBox to remove the purchased items from the GroceryList.

Variations:

- Shopper purchases items on the GroceryList from multiple stores.
- Shopper purchases items online and are delivered.

3.1.3.1.2.1.1.4 Cook & Eat Meal

1. Cook opens eRecipeBox and selects the RecipeCard(s) scheduled for that day's meal.
2. Cook views RecipeCard(s) and follows the instructions to combine the ingredients and cooks dishes (touch screen preferred).
3. Eaters (family members) eat the meal.
4. Cook surveys the Eaters on how they liked the dishes and updates the family rating for the corresponding RecipeCard(s).

Variations:

- If the Cook uses up a staple item while preparing a CookedDish, Cook adds the item to the GroceryList.
- No time to go to the store. Cook searches RecipeBox for RecipeCards containing ingredient(s) currently at home.

3.1.3.1.2.1.2 Admin Use Cases

Solution Model Actors: Administrator

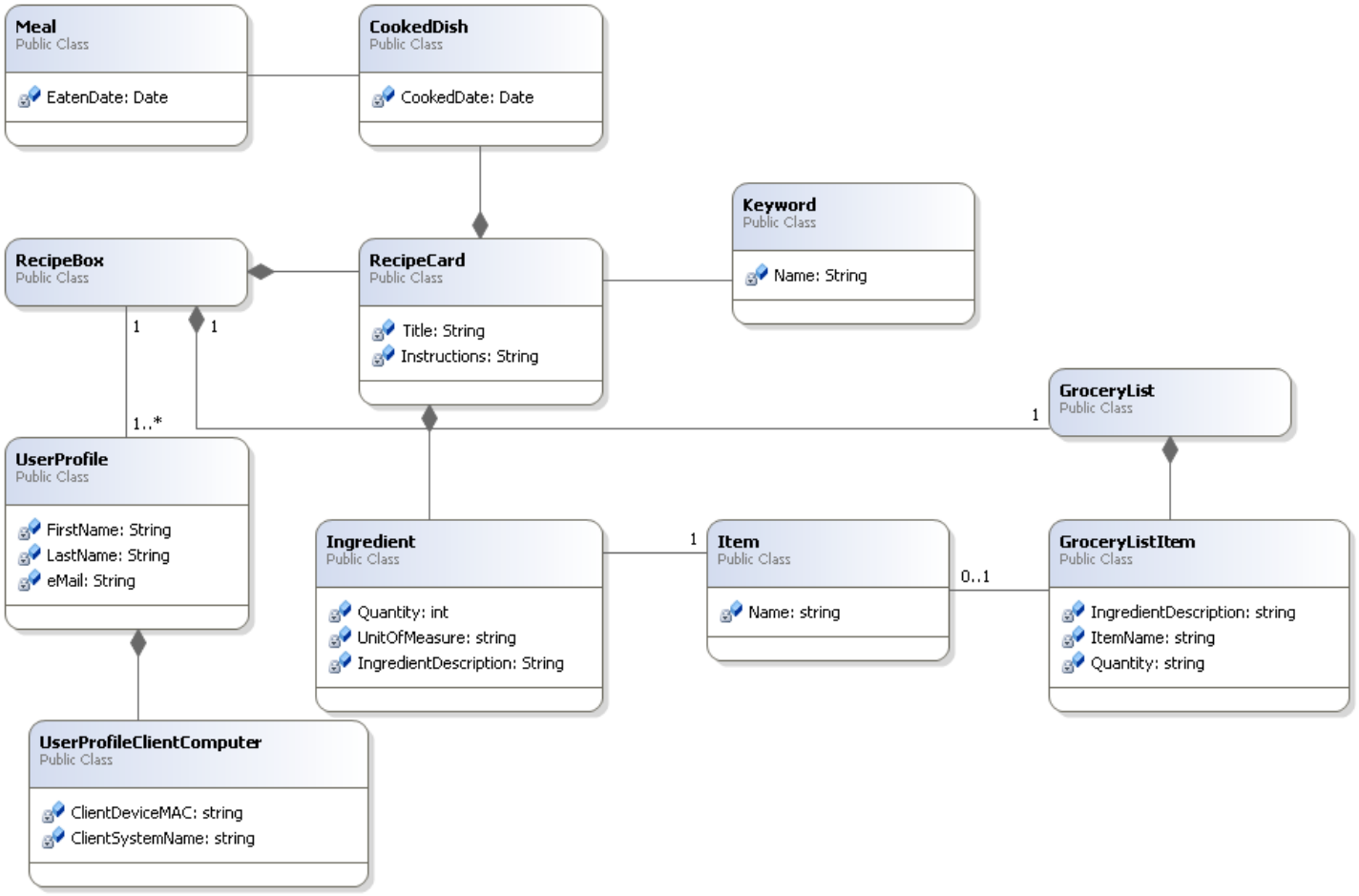
3.1.3.1.2.1.2.1 Maintain UserProfile

1. Administrator adds/deletes UserProfile and grants access to their RecipeBox

3.1.3.1.2.1.2.2 Backup / Restore DataStore

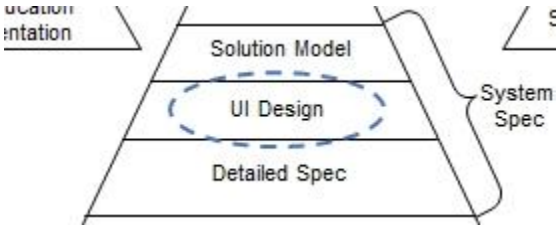
1. Administrator backs up all data from Admin form in eRecipeBox. Must be fast and easy.
2. Administrator restores from a prior backup

3.1.3.1.2.2 To-Be Solution Class Diagram



The solution digitizes the As-Is solution **RecipeCards** which are stored in **RecipeBoxes**. **RecipeCards** contain a list of **Ingredients**. Each **Ingredient** includes quantify of a food **Item** and any preparation instructions. Users can associate **Keywords** (eg, "Main Dish") to RecipeCards to make them easy to find. Each **RecipeBox** is associated with a single **GroceryList** containing **GroceryListItems**. Each **GroceryListItem** could be optionally populated by an Item on a **RecipeCard** or merely entered by the user. Each user has a **UserProfile** that is associated with one and only one **RecipeBox**. Multiple **UserProfiles** may have access to the same **RecipeBox**. The system captures **UserProfileClientComputer** information for each client PC **UserProfile** connects to the server.

3.1.3.2 UI Design



The UI Design presents the overall vision and experience paradigm for the end users. It contains four sections.

- System User Personas
- UI Conceptual Framework & UI Standards
- UI Navigation Map
- Bodies of Business Objects

3.1.3.2.1 System User Personas

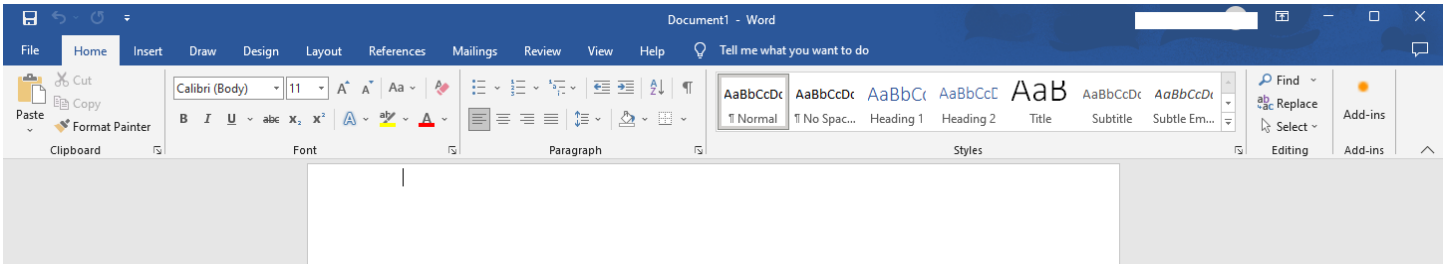
All system actors (Meal Scheduler, Shopper, Cook, Family members) are proficient with Microsoft Excel, Word, and Outlook. They are also able to process emails on their mobile phone. Administrators are more technical and capable of creating and restoring backups.

3.1.3.2.2 UI Conceptual Framework & UI Standards

UI must be simple and intuitive. Minimum training required, if any.

Assume users are comfortable with MS Office (especially Outlook, Word and Excel), so mimic Office design to minimize training. More specifically, use a single main window that contains ribbon forms used to CRUD business objects. Open a business object form that occupies the entire main window. All buttons are located in the Ribbon so user knows where to look for commands. Paradigm- Search, then view/edit the business object and Save, Cancel or Save & Close it.

Sample Microsoft Word ribbon form for reference.



Buttons in the ribbon are grouped based on the business object they operate – RecipeBox, RecipeCard, Ingredients, Calendar, GroceryList.

Window management: Follow a simple web-inspired paradigm – Opening a new form pushes it on the form stack. Closing a form pops the stack and returns to the previous form in its previous state.

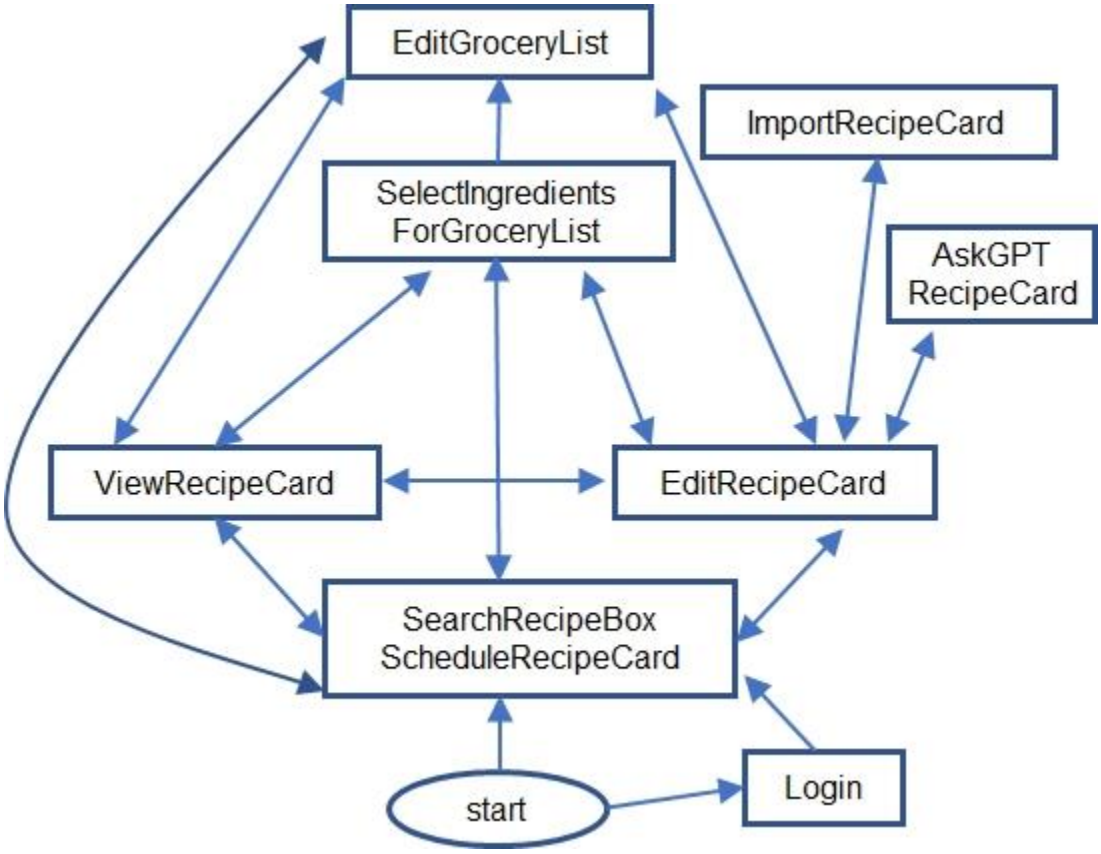
UI must support touch which is important when viewing the RecipeCard when cooking in the kitchen.

UI responses must be super responsive and efficient – less than .5 sec average response time.

To minimize switching hand between keyboard and mouse, support shortcut keys (Ctl-c, Ctl-v, etc.)

3.1.3.2.3 UI Navigation Map

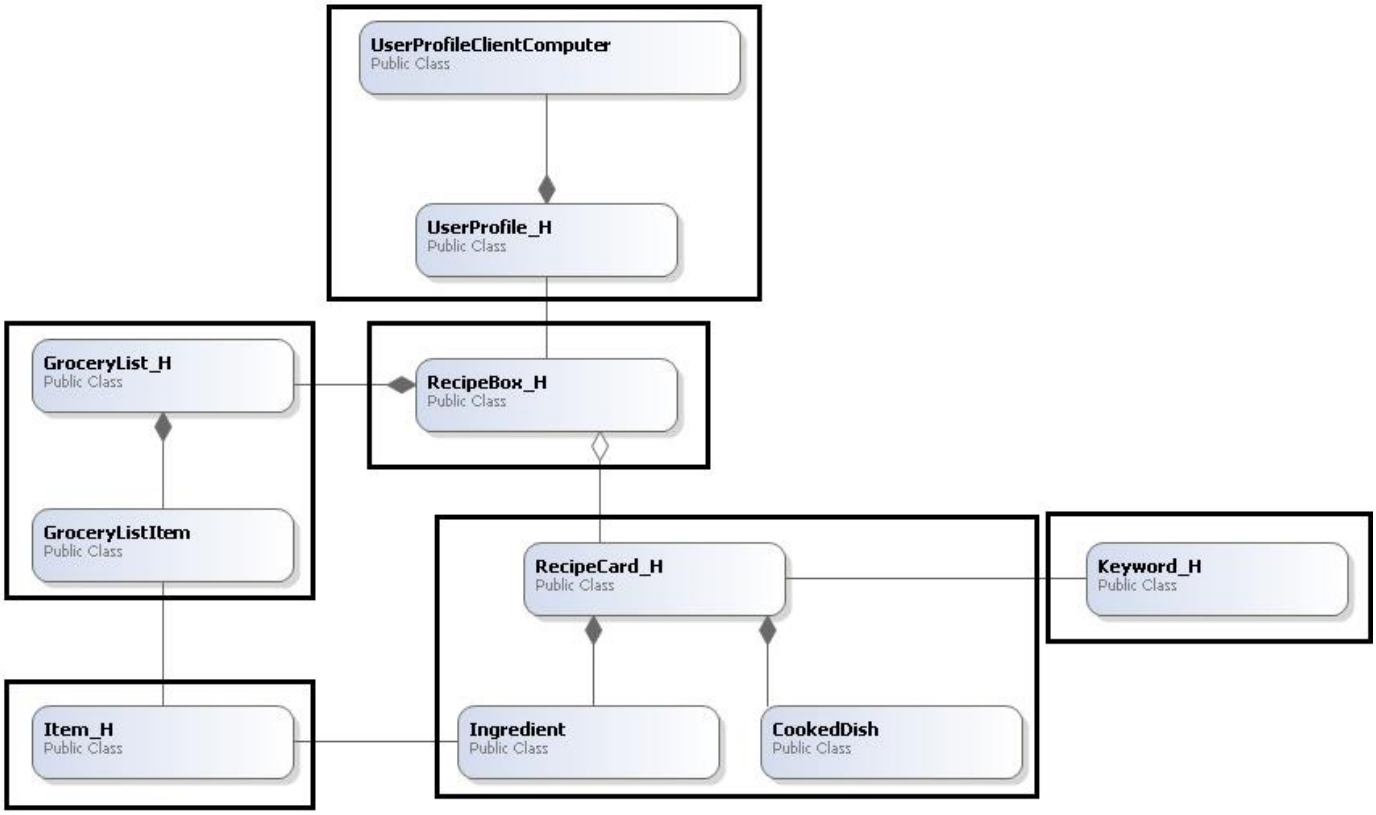
eRecipeBox forms and their navigation flow.



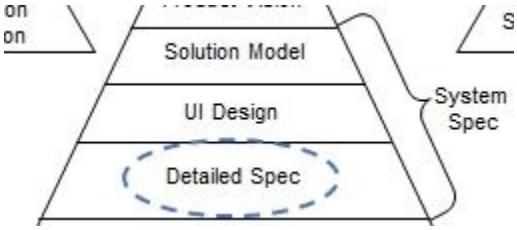
eRecipeBox prompts the user for UserProfile info upon the first login. User credentials are saved on the PC client so the user doesn't need to provide login data on future logins.

3.1.3.2.4 Bodies of Business Objects

Users need to understand what data is and is not being saved when hitting the save button, i.e., the scope of the unit of work being committed. eRecipeBox saves bodies of business objects in the DataStore in a single transaction. Each body has a Head, indicated by the _H suffix in the following class diagram. Bodies are fetched and saved by their heads. Business domain model instances are created, updated, and deleted using the following scopes. For example, saving a RecipeCard head also saves its CookedDishes, Ingredients and its references to Keywords.



3.1.3.3 Detailed Spec



Detailed System requirements are structured as follows –

- Functional Requirements
  - Business Rule Rqts
  - UI Forms - Detailed Rqts
  - Business Automation Rqts
  - Import/Export/Print Rqts
  - Reports
  - Security Rqts
  - External System Interfaces
  - System Administration/ Operations Rqts
  - Migration Rqts
- Non-Functional Requirements
  - Design Constraints/Deployment/ Installation/ Configuration Rqts
  - Performance/Load/ Concurrency Rqts
  - Scalability/Capacity Rqts
  - Robustness/Availability Service-ability/ Fault Tolerance Rqts
  - Internationalization Rqts
  - Product Lifespan Rqts

3.1.3.3.1 Functional Requirements

This section specifies detailed decisions that impact external actors as they use the system. Unlike system design, external actors *will* notice if we change these decisions - analogous to selecting paint colors when building a custom home.

3.1.3.3.1.1 Business Rule Rqts

The Solution Model class diagram specifies the business classes, their properties & relationships among each other. This section specifies the additional constraints and behavior the business classes must follow to ensure high-quality data. For example, an Account Number must be unique for each Account.

Note: *This section is automatically generated from annotations embedded in the code. Do not edit this next sentence.*

Business rule requirements are organized by their associated business class, one subsection for each.

3.1.3.3.1.1.1 CookedDish Business Rules

CkdDshDtNoDup Don't add duplicate CookedDish dates for a RecipeCard. When editing a cooked dish to a date that already exists, process this as a no-op.
CkdDtRqrd CookedDate is required.

3.1.3.3.1.1.2 GroceryList Business Rules

GLEmlSrt Sort grocery list by Item Name when emailing the grocery list.

3.1.3.3.1.1.3 GroceryListItem Business Rules

ItmRqrd Item is required.

3.1.3.3.1.1.4 Ingredient Business Rules

IngPrsItmDscr Parse ingredient string into its parts (Qty, UoM, Item Description) when importing. Heuristic: Look for a recognizable unit of measure and use it as a parsing anchor.
IngSctnHdr
Bold a section header that is determined by -
1. No quantity, UoM or Item.
2. Section name in ItemDescription ends with ":" (eg, Add and Simmer:)
IngSggstItem
Attempt to automatically select an Item based on ItemDescription. Heuristic -
1. Convert all words to their singular form (eg, eggs to egg).
2. Search all Items for an exact match.
2. If no exact match, search (contains) Items.
3. If there is only 1 match result whose Name length is 5 or greater, auto-select it.
IngTrnslDcmlQty Auto-translate decimal quantities into fractions. eg, .25 is auto-translated to 1/4
IngUsrOrdr Allow user to set the display order of ingredients.
ItmDscrptnRqrd ItemDescription is required.
SrtOrdrRqrd SortOrder is required.

3.1.3.3.1.1.5 Item Business Rules

ItmClnCapNm Auto-clean and capitalize all Item Name words. eg, Red Onion.
NmRqrd Name is required.

3.1.3.3.1.1.6 Keyword Business Rules

KywdClnCapNm Auto-clean and capitalize all Keyword words, e.g. Side Dish.
NmRqrd Name is required.

3.1.3.3.1.1.7 RecipeCard Business Rules

CrtdDtRqrd CreatedDate is required.

Don 't add duplicate CookedDish dates for a RecipeCard. Adding a date that already exists is a no-op.
ItmDescAtoFll When saving RecipeCard, if an ingredient has a null ItemDescription, set it to the ItemName (because ItemDescription is required).
ItmSggst When saving RecipeCard, if an ingredient has a null Item, attempt to auto-suggest an item.
MyRtgRng MyRating score range: 1-10.
RecpCrdKwdNoDup Don't add the same keyword twice to a RecipeCard. Adding an existing keyword is effectively a no-op.
TtlRqrd Title is required.
TtlUnq Title must be unique for each RecipeCard within the context of a RecipeBox.
WldLkTTryFlgRqrd WouldLikeToTryFlag is required.

3.1.3.3.1.1.8 UserProfile Business Rules

FrstNmRqrd FirstName is required.
HmZpCdRqrd HomeZipCode is required.
LstNmRqrd LastName is required.
PrsnlCllNmbrRqrd PersonalCellNumber is required.
PrsnlEmlRqrd PersonalEmail is required.

3.1.3.3.1.1.9 UserProfileClientComputer Business Rules

ClntSystmNmRqrd ClientSystemName is required.
InstlDtRboxApp Store date and time that eRecipeBox was installed on the client computer
LstLgnDtCC Store the most recent date and time that user logged in from each client computer
NmbrLgnsCC Log number of times the user has logged in from each client computer

3.1.3.3.1.2 UI Forms - Detailed Rqts

This end user education video demonstrates eRecipeBox UI Forms and their navigation.

3.1.3.3.1.2.1 Common UI Detailed Rqts

Button order for CRUDing business objects- New, View, Edit, Delete, Refresh.
Auto-trim all text entered by the user - trim leading/trailing spaces and set to Null if all whitespace.
Enforce text data input to its max length allowed in the DataStore.
Mask enforce text input when possible.

Note: *This section is automatically generated from annotations embedded in the code.*
The remaining subsections specify the detailed requirements for each UI form, one subsection for each form.

3.1.3.3.1.2.2 Authenticator

LginCachEmlId Cache the user's email in a file (encrypted) on the client PC so eRecipeBox doesn't need to prompt the user for it on subsequent logins.

3.1.3.3.1.2.3 EditGroceryList

1EdtGLFrm
EditGroceryList behavior -
Enter GroceryListItem and Add to the GroceryList.
GroceryListItems can be edited inline in the grid (to modify qty, add preferred brand, etc.)
To Delete GroceryListItems, select rows (1st column) and DeleteItems.
Once complete, Email GroceryList to the email address in the user's UserProfile.
GLEmlHTML Generate an HTML table when emailing GroceryList.
GLEmlOAth Email GroceryList to the email address in the user's UserProfile.
GLItmAtoStItm If Item is empty and ItemDescription has a value, set Item to the same value.
GLItmGrdDspCnt Display number of GroceryListItem rows in the status bar.
GLItmInptCntns GroceryList Item LookupEdit shows matches that contain the entered text.
GLItmsBtnEnbl Disable all grid buttons if the grid is empty.

GLItmsBtnGrdFcs Only enable grid buttons when GroceryListItems grid has focus
GLItmsInptSggstLst Populate GroceryListItem LookUpEdit suggested input with all Items, however user can add anything to the GroceryList (e.g., light bulbs).
GLItmsSrtOrdr Sort GroceryListItems by Item so the same Items are grouped together.
ShCut Shortcut Keys: Alt-c (save and close), Ctl-s (save), Delete key (delete), Alt-e (email GList), Alt-l (clear all GLItems)

3.1.3.3.1.2.4 EditRecipeCard

1EdtRcpCrdFrm
EditRecipeCard behavior -
User creates/imports/modifies/deletes a single RecipeCard with this form.
Users can only import into a new, empty RecipeCard.
BtnAdCkdDt Add cooked date
BtnAdCkdDtTdy Add today as cooked date
BtnAdIngs2GL AddToGroceryList button: Populate SelectIngredientsForGroceryList form with all ingredients in the RecipeCard. User removes ingredients not needed, then adds the remaining to the GroceryList.
BtnEdtTtlUrl Edit Title button allows user to edit the title and the RecipeCard SourceURL.
BtnImprtEnbl Enable import buttons only when creating a new RecipeCard. This prevents accidently importing on top of an existing RecipeCard.
CkdDtDsplyDsc Display CookedDates in descending order.
GrdBtnsEnbl Enable grid buttons (insert, move up/down, etc.) only when grid has focus.
GrdUpDnBtnsEnbl Up/Down buttons only valid for ingredients
ImprtAllRcps Prompt user for a recipe URL in AllRecipes.com and automatically import it.
ImprtGptRC Prompt user for GPT prompt. Import RC from GPT.
ImprtTxtRC Parse RC text and import it into a new RC
IngBstFtCls Auto-best fit columns in Ingredients grid.
IngMltSlct Ingredients grid supports multiple row selection.
IngSctnHdrBld Bold section headers, indicated by ItemDescription that ends in a colon ':' and other fields empty (eg, Toppings: Spices: Dressing:)
ItmCrNwSvRC Allow new Items to be added to lookupedit. When adding new Item, first autosave the RecipeCard (without the new Item), then add new Item, then add reference to the new Item.
KywdAdNw Allow users to add new Keywords by typing a new term into the Keyword lookupedit. To ensure proper business object consistency, force saving current RecipeCard before adding a new Keyword
KywdDsplyAsc Display Keywords in ascending order.
RCSchCkdDt ScheduleRecipe button: Prompt user for a date to schedule the recipe. Once scheduled, set focus to the CookedDishes tab so the user sees the new date.
ShCUT Shortcut keys - Alt-c (save and close), Ctl-s (save), Alt-v (view recipeCard), Alt-i (import recipeCard), Alt-s (schedule recipe), Alt-r (refresh), Insert (insert row in grid), Ctl-d or Ctl-delete (delete row in grid), Alt-g (open groceryList), Alt-a (add ingredients to GList)
TtlTxtHyptxt Title is displayed either as a readonly text label or readonly hyperlink, depending whether there is a URL. To edit these, user clicks "Edit Title" button. System presents a pop-up dialog to set both the title and URL.

3.1.3.3.1.2.5 ImportRecipeCard

Importing a RecipeCard is performed in three steps:
1 . Create a sequence of lines containing RecipeCard property markers (eg "Title:"), followed by their property values.
2 . Parse the individual parts of each Ingredients into its properties. Use <tab> as a delimiter.
3 . Import the parsed text lines into a RecipeCard by setting RecipeCard's properties with the text.
Shortcut keys - Alt-p (parse raw RecipeCard text), Alt-i (import parsed text into RecipeCard and close)

3.1.3.3.1.2.6 RecipeBoxMDIParent

AppClsPrgSftDel Purge the soft deleted objects in the database when closing the app.

3.1.3.3.1.2.7 SearchRecipeBox

1SrchRcBxFrm
SearchRecipeBox behavior -
Enter FilterTerms, one at a time, and click Filter.
Each additional FilterTerm is AND-ed with the previous FilterTerms.
System displays results in the grid and row count in status bar.
User can update MyRating and Flagged directly in the grid without opening the RecipeCard.
User can also edit/delete the last CookedDate or add a new CookedDate directly in the grid without opening the RecipeCard.
Drag and drop filter results grid RecipeCard onto Calendar to schedule RecipeCard.
Also, can Copy grid RecipeCard onto Calendar to schedule RecipeCard.
Drag and drop events within Calendar to reschedule RecipeCard(s).
Also, Cut/Paste events within Calendar to reschedule RecipeCard(s).
Also, Copy/Paste events within Calendar to create additional cookedDishes for these RecipeCard(s).
Calendar: Hold CNTL key to multi-select Calendar events.
Calendar: Hold SHFT key to select all events between two days (all events on all these days)
User can also dragNdrop RecipeCard(s) events from Calendar onto the RHS (months) navigation calendar.
BtnAd2GL
AddToGroceryList button:
1. Populate SelectIngredientsForGroceryList form with all Ingredients of all selected RecipeCards (grid or calendar, whichever has focus).
2. User deletes the Items not needed, then adds the remaining to the GroceryList.
BtnAdBkUpDB Admin.BackupDB button: Backup DataStore schema definition and all data to a set of files in a folder on the PC.
BtnAdRstrDB Admin.RestoreDB button: Restore DataStore schema definition and all data from a backup folder. Prompt user for the folder.
BtnDelRecCrdCnfm Prompt user for confirmation before deleting RecipeCard.
BtnDelRecCrdEnbl Enable delete RecipeCard button only when grid is focused on a RecipeCard; never when Calendar has focus.
BtnFltr
Filter my RecipeBox by OR-ing these fields with each FilterTerm - Title, Notes, Description, Instructions, SourceIndividualFirstName, SourceIndividualLastName, Ingredient.ItemDescription, Ingredient.Item.Name, Keyword.Name
Each FilterTerm is AND-ed with the previous FilterTerm(s).
eg, Easy AND Mexican. Find all RecipeCards that have 'Easy' in one of the fields AND 'Mexican' in one of the fields.
BtnVwEdRecCrdEnbl Only enable View/Edit/DeleteRecipeCard if we have focus on a valid RecipeCard.
ClndrClckShft Consistent with Exel's grid, Shift+Click selects all recipes between two dates and Cntl+Click union selects clicked events on the calendar
ClndrClpbrd
Ctl-c copies (Ctl-x cuts) to Windows clipboard and to app clipboard.
Ctl-v, if copied, creates new CookedDishes at selected day
Ctl-v, if cut, moves CookedDishes to the selected day
ClndrCtlEntr Ctl-Enter opens EditRecipeCard iif one RecipeCard is selected in the Calendar.
ClndrDblClk Double click opens EditRecipeCard iif one RecipeCard is selected in the Calendar.
ClndrDel
Deleting (Delete key) selected RecipeCard(s) in the calendar deletes the corresponding CookedDish(es) for the selected RecipeCards.
Note: To prevent accidental RecipeCard deletion, user cannot delete a RecipeCard from the Calendar.

ClndrEntr
Enter opens ViewRecipeCard iif one RecipeCard is selected in the Calendar.
ClndrFcusGrd Set focus to the scheduled RecipeCard in the grid after drag and drop to the Calendar.
FltrTrmInpt
LookupEdit behavior -
a) Enter key applies the selected drop down text (or new text) as a new filter term and runs the filter.
b) Start to type (shows autocompletion), then tab. Accepts the autocomplete.
FltrTrmInptCntns FilterTerm LookupEdit shows matches that contain the entered text.
FltrTrmInptLkUps List of suggested FilterTerms in the LookupEdit contains all Keywords, Items and RecipeCard.Titles - no duplicates.
GrdCkdDtEdtStFcs Set focus to the edited RecipeCard after adding/editing LastCookedDate in grid.
GrdClsRCFrmFcs Set focus to the previously Viewed/Edited RecipeCard in the grid after closing the form.
GrdCtlEntr Ctl+Enter key in grid launches EditRecipeCard.
GrdDblClk Double click opens the EditRecipeCard form on clicked RecipeCard in the grid.
GrdEdtFlg Toggle flag in the grid
GrdEdtLstCkdDt edit, add, delete last cooked date in the grid
GrdEdtMyRtng Edit myRating in the grid
GrdEntr Enter key in grid launches ViewRecipeCard.
GrdLstCkdDtDelScr Scroll to the top of the grid after deleting a last cooked date from the search results grid.
GrdTtlHyprLnk Click on title to open the source URL in a browser.
ShCut
Shortcut keys: Alt-c (Close app), Alt-v (viewRecipeCard), Alt-e (editRecipeCard), Alt-n (createNewRecipeCard), Alt-g (editGroceryList), Alt-a (addToGroceryList), Alt-f (Filter), Alt-L (clear input), Alt-i (move cursor to the FilterTerm (input)), Alt-t (Today),
Shft-e (Edit MyRating)
Shft-e (Edit LastCookedDate), Shft-a (Add New CookedDate), Shft-d (Delete LastCookedDate)
Shft-e (Edit LastCookedDate), Shft-a (Add New CookedDate), Shft-d (Delete LastCookedDate)
Shft-e (Edit MyRating)

3.1.3.3.1.2.8 SelectIngredientsForGroceryList

1SlctIng4GLFrm
SelectIngredientsForGroceryList behavior -
eRecipeBox populates SelectIngredientsForGroceryList with all ingredients from all selected RecipeCards and displays them in a grid.
User removes items not needed.
User clicks Add Items to Grocery List.
GrdMltSlct Ingredients grid is read only, multiselect rows.
GrdSrtByItm Sort Ingredients by Item so Items are grouped together.
IngIgnrSctnHdr Do not add section headers (eg, Spices:) to the GroceryList.
ItmStIfNull If Item is null, set it to ItemDescription since grid is sorted by Item.
ShCut Shortcut keys: Delete, Ctl-D (Delete rows in grid), Ctl-L (clear grid selection), Alt-A (Add Items to GL), Alt-C (Cancel)

3.1.3.3.1.2.9 ViewRecipeCard

1VwRecpCrdFrm
ViewRecipeCard behavior -
ViewRecipeCard supports cooking a recipe - Read only mode.
User can move the splitter to resize the 2 panels.
User can Schedule & Delete the RecipeCard.
User can also Open the GroceryList or Add the Ingredients to the GroceryList from this form.
IngSctnHdrBld Bold section headers, indicated by ItemDescription that ends in a colon ':' and other fields empty.

Shortcut keys - Alt-c (Close), Alt-e (editRecipeCard), Alt-d (deleteRecipeCard), Alt-g (editGroceryList), Alt-a (addIngredientsToGroceryList), Alt-s (scheduleRecipeCard),

3.1.3.3.1.3 Business Automation Rqts

[N/A for MVP 1.0]

3.1.3.3.1.4 Import / Export / Print Rqts

Import RecipeCard from structured text, AllRecipes.com, or GPT.

3.1.3.3.1.5 Reports

[N/A for MVP 1.0]

3.1.3.3.1.6 Security Rqts

Security requirements are covered in four sections–

- User Identification & Authentication Requirements – Identify user and verify they all granted access to the app
- Authorization – Only allow user access to functions and data that they are granted access
- Data Privacy Requirements – Protect sensitive data (e.g., passwords, sensitive data, such as SSNs)
- User Audit Requirements – Log user data access and user operations

3.1.3.3.1.6.1 Identification & Authentication Rqts

A user is identified by their email address and their eRecipeBox client PC MAC address, stored in their UserProfile. Following are the authentication use cases.

User authentication prerequisites:

- User installs eRecipeBox app on their Windows PC.
- Administrator creates a UserProfile in the DataStore with the user's email address.

Authenticate User– First login from client computer

1. User launches eRecipeBox app
2. eRecipeBox prompts for all UserProfile fields on the Login form.
3. User supplies UserProfile info
4. eRecipeBox caches the email address on the client PC associated with the Windows User Name
5. DataStore service verifies there is an existing UserProfile with the email address, rejects login if not.
6. eRecipeBox updates UserProfile with name, phone & zipcode info and stores the Windows user name, system name and MAC address in the ClientComputer DataStore.

Authenticate User– Subsequent logins from client computer

1. User launches eRecipeBox app
2. eRecipeBox reads the email from cache, bypassing the need for the Login form
3. DataStore service verifies there is a UserProfile with the same email address and MAC address, rejects login if not.

3.1.3.3.1.6.2 Authorization

Data Authorization

Each UserProfile is associated with one RecipeBox and has full access to their RecipeBox, its RecipeCards and its associated GroceryList. Multiple UserProfiles can access the same RecipeBox.

Functional Authorization

Uers can perform all CRUD functions on RecipeCards in their RecipeBox. Users can create new Items and Keywords.

3.1.3.3.1.6.3 Data Privacy Rqts

Encrypt user identity (email address) on the ClientComputer.

Encrypt all messages between the client and app server over HTTPS.

Encrypt all passwords that reside in files on the server, such as a password in a DataStore connection string.

3.1.3.3.1.6.4 Auditing Rqts

Log eRecipeBox application install date, last login date and number of logins with each UserProfileClientComputer.

3.1.3.3.1.7 External System Interfaces

Import a structured RecipeCard from www.allrecipes.com or GPT API.

Support speech recognition using Azure services when prompting for GPT.

Send GroceryList to the user's email address using Google email OAuth 2.0.

3.1.3.3.1.8 System Administration / Operations Rqts

Provide backup & restore of all eRecipeBox data in its DataStore from the eRecipeBox client for 1-tier and 2-tier deployments.

Create/delete UserProfile with its RecipeBox and GroceryList.

3.1.3.3.1.9 Migration Rqts

The customer is responsible for importing legacy recipe cards into eRecipeBox once deployed to production.

3.1.3.3.2 Non-Functional Requirements

This section specifies detailed decisions on how well the system performs its functions, ones that external actors will notice. Whereas functional requirements specify the static & dynamic behavior of the system (i.e., think nouns and verbs, "the System does X"), non-functional requirements qualify and quantify the functional requirements (i.e., think adjectives and adverbs, "the System does X fast/reliable/maintainable" or "Data is small/medium/large/colossal"). This is analogous to specifying the expected annual utilities/annual maintenance cost/lifespan when building a custom home.

3.1.3.3.2.1 Design Constraints / Deployment / Installation / Configuration Rqts

Must run on the family's existing PCs - Windows 10 or 11; desktop or laptop PC with or without touch screen.

3.1.3.3.2.2 Performance / Load / Concurrency Rqts

Sub-second average response time for all simple UI interactions. Average filter RecipeBox shall be less than 2 seconds.

Support 1,000 concurrent users with sub-second average response time.

Optimistic concurrency lock on each Body of Objects. Notify user if Body has been modified by another user and refresh the Body to current version before user makes their edits.

3.1.3.3.2.3 Scalability / Capacity Rqts

eRecipeBox should support linear scalability up to 100,000 concurrent users. Support 1,000 concurrent users the first year, then double each year for the next 7 years.

MVP 1.0: Maximum 25,000 RecipeBoxes and similar number of UserProfiles. Each RecipeBox averages 300 RecipeCards totaling ~7.5M RecipeCards.

3.1.3.3.2.4 Robustness / Availability / Service-ability / Fault Tolerance Rqts

99% uptime. No more than 7 hours of scheduled service downtime per month, occurring between 2-4AM ET.

3.1.3.3.2.5 Internationalization Rqts

Support US English and US currency initially. However, plan to enhance the app to support all left to right languages starting in year 3.

3.1.3.3.2.6 Product Lifespan Rqts

eRecipeBox app life span is 15-20 years.

3.1.3.3.3 Build the Detailed Spec

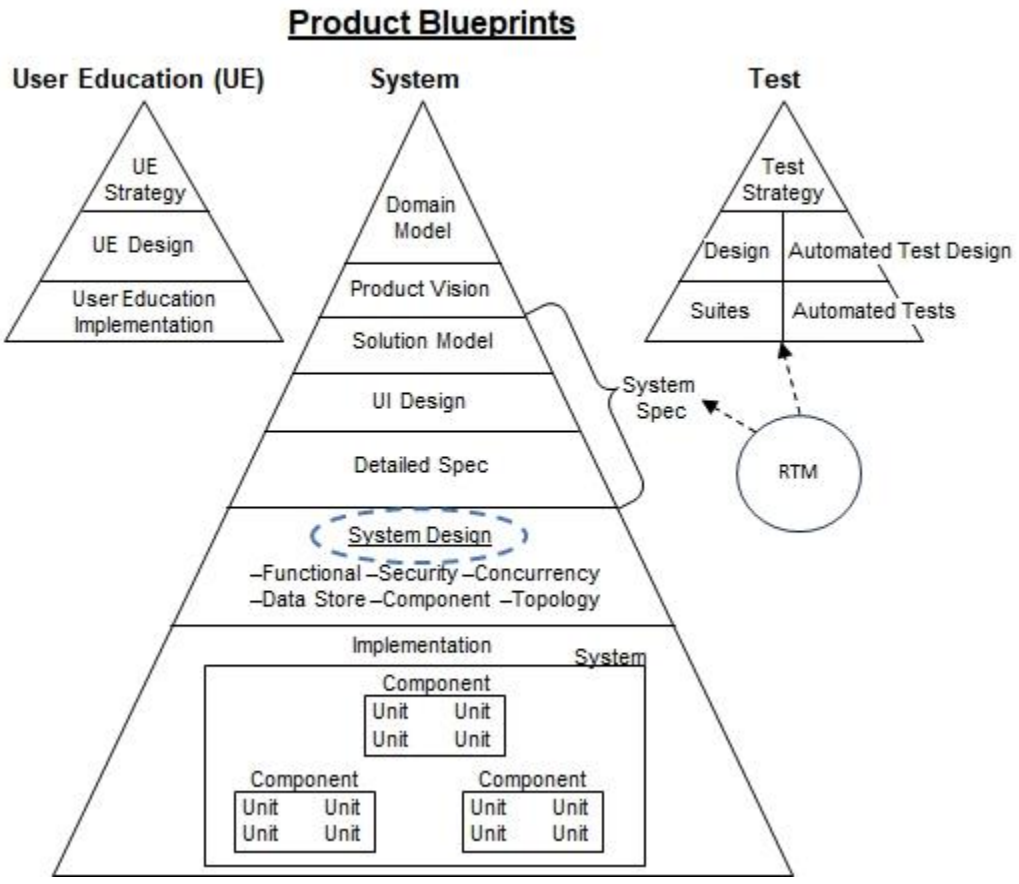
Business rule and detailed UI form requirements are embedded in source code. Follow these steps to add these two sections of the Detailed Spec.

1. Ensure \ eRecipeBoxSampleApp\ eRecipeBoxSystem\ eRecipeBox\ Documentation\ eRecipeBoxArtifacts.docx is closed.
2. Open eRecipeBox.sln
3. Set DocumentGenerator as startup project.
4. Run.

eRecipeBoxArtifactsFinal.docx is generated in the same folder. Word opens the generated doc.

3.1.4 System Design

The System Design is organized into six views at the same level of detail. Every architect needs to make many decisions solving various aspects of the solution. These views provide a logical, well-organized structure to record and communicate these decisions.



Functional Design– An object model of the core implementation classes. Narrates how the system logically implements the functionality specified in the solution model for a single system actor. Presents patterns for implementing common functionality throughout the application. This view isn't concerned with any aspects addressed in the other 5 views (e.g., security or multiple actors requesting services simultaneously).

Security Design– Presents how the system implements the security requirements- authentication, authorization, protecting secrets, and audit logging.

Concurrency Design– Replays the main scenario in the functional and security designs with multiple actors requesting services simultaneously. Illustrates how the system serves these requests with satisfactory response times and ensures data integrity. Describes how the system scales to support increasing user populations and demands, often with parallelism and scaling up & out.

DataStore Design– Presents the logical and physical representation of all persistent data– databases, files, etc. It explains how object model classes in the functional design map to their physical DataStore representation.

Component Design– Presents the packaging of software code into libraries and their dependencies. Both developed and procured.

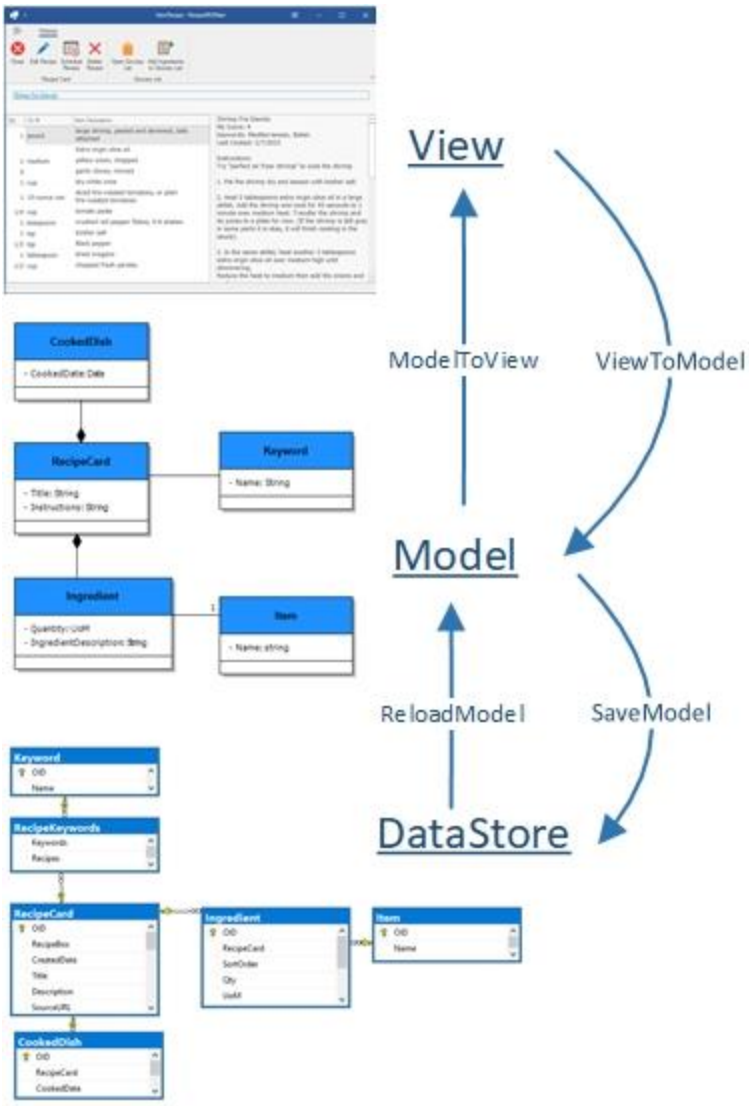
Topology Design– Presents the logical and physical structural System deployment- Service Design and Network Design.

3.1.4.1 Functional Design

The Functional Design describes the internal software design for the core functionality. To minimize complexity, its scope doesn't include design decisions documented in the other five views (e.g., concurrency, security, etc.). This section also describes all design patterns in the System architecture.

3.1.4.1.1 Model-View Design

eRecipeBox follows a simple View/Model/DataStore design; more specifically, a View (WinForms), Model (C# business objects), DataStore (relational database) design. Persistent data from the DataStore is loaded to the client PC and reconstituted into Solution Model class instances. UI Views present the Model data to the end user. User modifies the data in the View, and the process is reversed to persist the changes back to the DataStore (ViewToModel, then SaveModel). Data binding between the view and model is leveraged as much as possible.



==========

Sidebar: Experienced developers are probably familiar with the [MVVM design pattern.] MVVM introduces another layer of classes (ViewModels) between View and Model (IMHO, a better name would be VVMM). eRecipeBox's design (and WinForm app designs in general) encapsulates binding, command and event handler logic in a code behind file of the view class. MVVM proponents assert MVVM designs are better because testers can write API tests against the VM classes. I've personally never observed any project do this successfully because emulating View behavior for realistic business scenarios is a huge endeavor, difficult and error-prone. In my experience, developing test scripts (manually or record & playback) against the View is more productive. eRecipeBox chose this approach for its automated tests. In addition, View and View-Model are very tightly coupled, so in many cases, any benefits of introducing another class layer are outweighed by the extra complexity.

To map eRecipeBox's View/Model/DataStore design to MVVM –

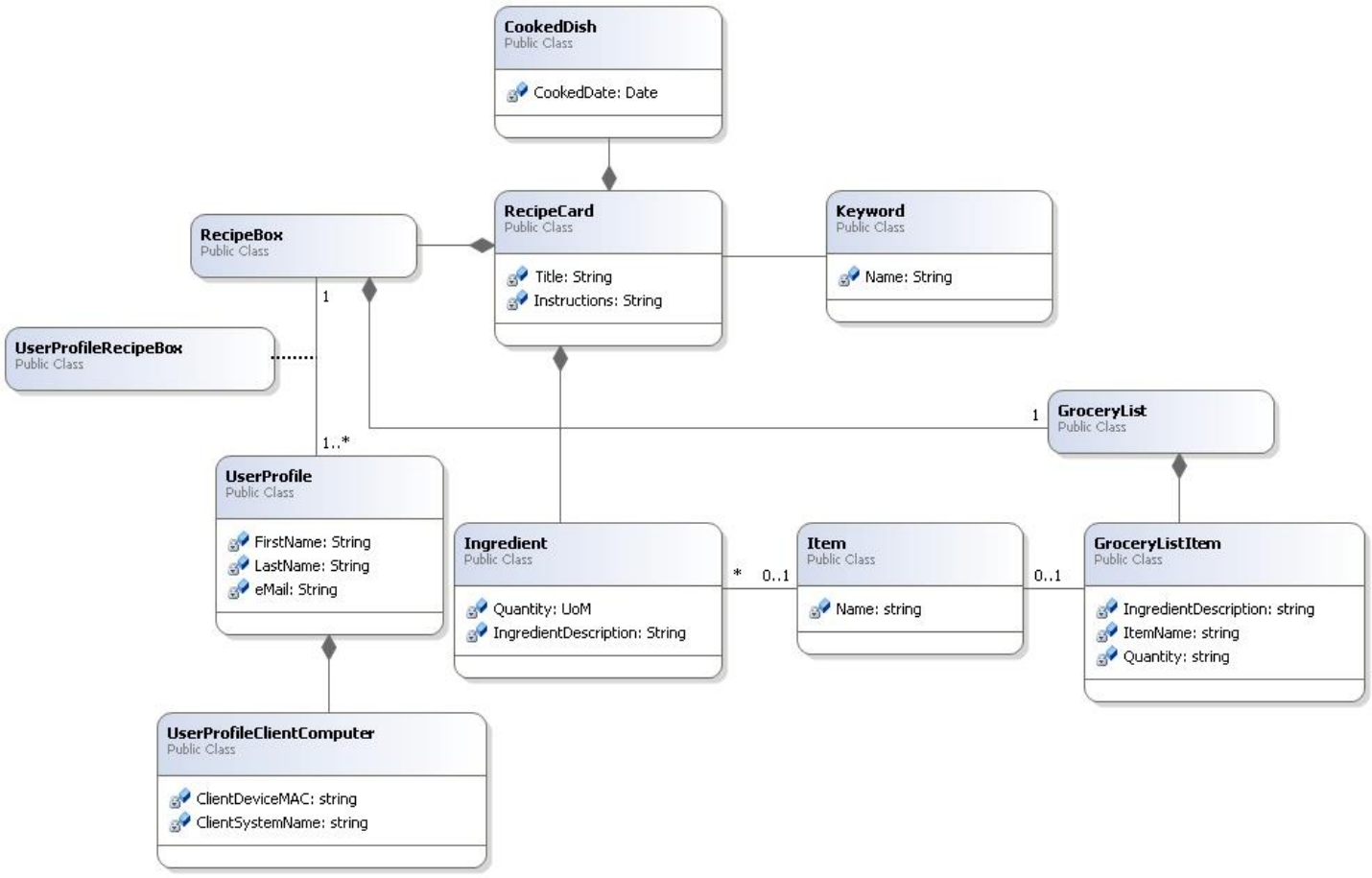
- <form>.designer.cs and <form>.resx map to View
- <form>.cs and <form>Helpers.cs map to View-Model
- RecipeBoxSolutionModel.<solution class>.cs map to Model

Once one masters a View-Model design, it's easy to transition to an MVVM design. I chose this simpler design because:

- KIS - MVVM adds unnecessary complexity to the design for this solution.
- Simpler designs are a better instructional tool.
- Simpler designs are easier to maintain.

==========

3.1.4.1.2 Implementation Model Design



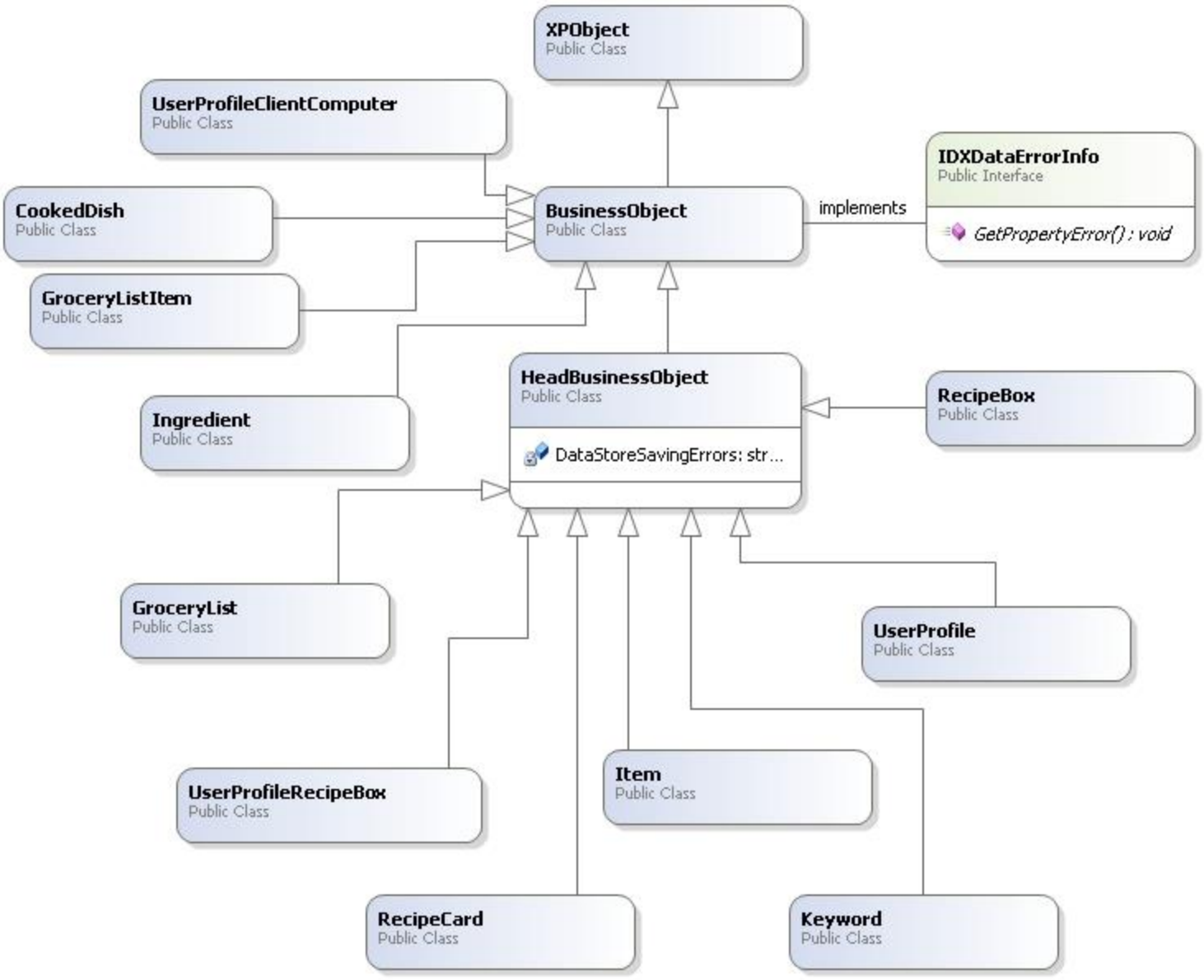
The implementation classes match the Solution Model except for a couple of minor differences -

- Do not persist Meal. Meals are merely a set of CookedDishes (from the same RecipeBox) on the same day.
- In anticipation of granting users access to multiple RecipeBoxes post-MVP, add an association class – UserProfileRecipeBox. For MVP 1.0, there is only one association for each UserProfile/RecipeBox pair.

Persist model classes in a database DataStore using DevExpress' XPO ORM product. If you are new to XPO and ORMs, see Hello World Console App with XPO and .NET | eXpress Persistent Objects | DevExpress Documentation and XPO/Tutorials/WinForms/Classic at master · DevExpress/XPO (github.com). DevExpress' ORM class implementation pattern is very straight forward and easy to learn. Review the model classes in RecipeBoxSolutionModel project.

3.1.4.1.3 Business Object Design

Saving bodies of business objects is supported through a couple of base classes- BusinessObject and HeadBusinessObject.



All BusinessObject classes enforce business rule checks following DevExpress' IDXDataErrorInfo design. The DevExpress UI framework calls GetPropertyError during form validation against controls bound to business objects – as a result; we encapsulate business rules checks in a single point of code, regardless of how many different places a business object is accessed throughout the application.

HeadBusinessObjects (and their bodies) are saved and deleted using XPO's UnitOfWork.Commit (see DataStoreUtils\XpoService). Any DataStore commit errors are saved in DataStoreSavingErrors in the HeadBusinessObject – keyed off the property raising the error. These errors are later returned in GetPropertyError() and displayed as error messages in the CRUDBusinessObjectForm. For an example, see how "each RecipeCard.Title within a RecipeBox must be unique" is enforced in the code.

3.1.4.1.4 Versioning the Implementation Model

Foundation.ModelVersion is a single table in the DataStore that stores versions of the implementation model. Each time a change is made to the Model, add a call to RegisterNewVersion in RecipeBoxSolutionModel.ModelInfo. Example:

```
RegisterNewVersion(SchemaDefinition, DateTime.Parse("2023-04-21 12:45:00"), false,
"ClientDeviceMAC no longer unique. Too restrictive.");
```

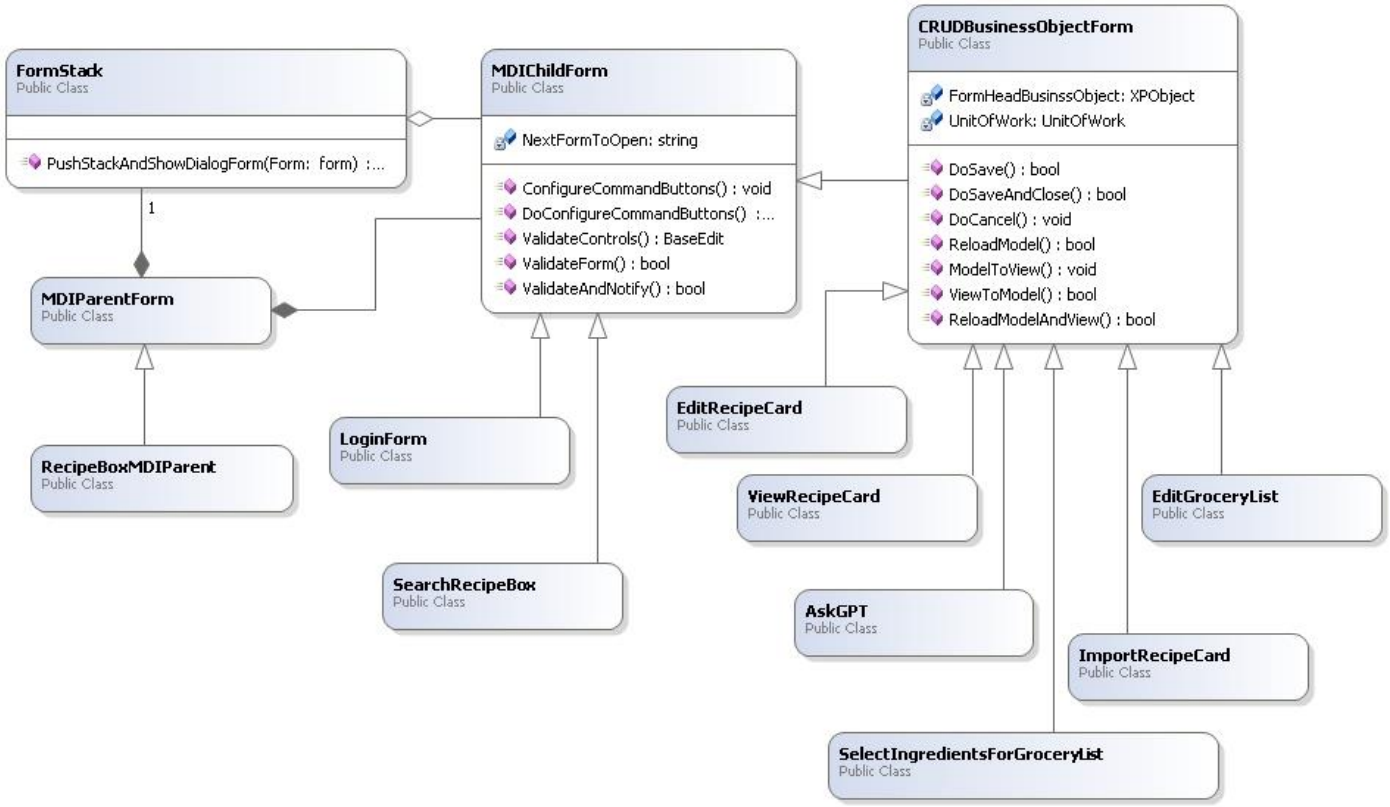
The app can compare its Model version with the DataStore and throw if they are not compatible.

3.1.4.1.5 Scalable, Serverless Design

All session state is maintained on the client app.

3.1.4.1.6 UI Framework Class Design

eRecipeBox is built on SimpleCRUDFramework that encapsulates all of the common behavior for CRUDing business objects using DevExpress' WinForms Ribbon-Based Application framework as well as implementing a simple FormStack for GUI window management. DevExpress' framework is implemented as a Microsoft Multiple Document Interface (MDI) application. BusinessObject subclasses DevExpress XPO.XPObject who serves as our ORM to synchronize the Model and the DataStore.



MDIParentForm serves as the base class for the single, eRecipeBox parent MDI form. MDIParentForm contains all MDIChildForms. MDIChildForms provides common logic for validating input on a form. CRUDBusinessObjectForm adds common logic for editing a body of business objects.

MDIChildForms are pushed and popped on a single FormStack to support the simple window navigation of returning to the previous form when closing a form. When pushing a new form on the stack and displaying it, minimize the current form to maintain its state.

3.1.4.1.7 Simple Search and Edit RecipeCard Use Case

1. User launches eRecipeBox app
2. The Main program –
    a. Creates a XpoDefault.DataLayer over a secure HTTPS WCF connection
    b. Runs a new RecipeMDIParent form

3. RecipeMDIParent_Load event handler Pushes a new SearchRecipeBox on the Application Form Stack and Shows it
4. SearchRecipeBox_Load event handler searches the RecipeBox (DoFilter()) with a null filter and displays the results
5. User browses results in the grid and scheduler which fires various events
6. User enters a filter term and clicks Filter button
7. Filter button event handler reruns DoFilter with the additional filter(s) which InitiateQuery to an xpInstantFeedbackView and binds it to XtraGrid filter results.
8. User browses the filter results, selects a RecipeCard in the grid (or scheduler) and clicks the Edit button
9. EditRecipeCard button event handler creates and Pushes a new EditRecipeCard form onto the Application Form Stack and Shows it
10. EditRecipeCard_Load event handler reloads the RecipeCard from the DataStore into the Model, syncs the model to the View and displays the form to the user
11. User edits the RecipeCard's ingredients, instructions, etc in the form then clicks the Save button
12. Save button event handler –
    a. Auto suggests Ingredient Items that are null, if possible
    b. Performs save (DoSave) which copies the view to model, performs validation and saves the model to the RecipeCard DataStore
    c. Reloads the model from the DataStore to load any changes other users may have made to the same RecipeCard body of BusinessObjects
    d. Syncs the model to the view (ModelToView) and displays the view to the user.
13. User clicks SaveAndClose button
14. SaveAndClose button event handler –
    a. Syncs the view to the model
    b. Determines no changes have been made to the model since the last save, so it closes the form.
15. EditRecipeCard's FormClosed event handler -
    a. Pops EditRecipeCard off the application form stack
    b. Invokes the FormClosedAction that was specified when EditRecipeCard was displayed (step #8 above) which
        i. Looks to see if a new form should be displayed (No), so it refreshes the current search results and sets focus to the previously edited RecipeCard, if possible.
    c. Fetches the top form on the application form stack (i.e., SearchRecipeBox) and displays it

The previous scenario serves as an example to illustrate the following design patterns.

3.1.4.1.8 Window Navigation Design

Follow the pattern described in the scenario. Namely,

1. Create a new form.
2. Call PushStackAndShowDialogForm passing the next form and a formClosedAction as parameters.
3. The formClosedAction can
    a. Interrogate any state (including the form's business object) in the pushed form to retrieve user activity while using the form.
    b. State also includes referencing the form's NextFormToOpen property. Use it to create and push and display the next form to the user. If NextFormToOpen is null, the current, invoking form (top of the stack) is displayed.

See SearchRecipeBox.ShowViewRecipeForm method and how it uses ShowNextForm in its formClosedAction as an example.

3.1.4.1.9 Edit (CRUD) a Body of Business Objects Design

All Editxxx forms are implemented as follows -

1. Inherit from CRUDBusinessObjectForm. Use data binding whenever possible, binding UI view controls to an Xpo.XPBindingSource. Set the XPBindingSource's Object ClassInfo property to the XPO head business object. Ensure all model (business objects) inherit from BusinessObject and the head from HeadBusinessObject.
2. Pass the OID of the business object in the form's constructor and invoke base(Oid). Null Oid indicates the form is creating a new body of BusinessObjects.
3. Ensure form's Load event handler calls ReloadModelAndView
4. <<User edits data in the view>>
5. <<User clicks Save, Cancel, Delete or Close button>>
6. Ensure Save button event handler calls DoSave
7. Ensure Refresh button event handler calls ReloadModelAndView
8. Ensure SaveAndClose form button event handler calls DoSaveAndClose
9. Ensure Cancel form button event handler calls DoCancel.
10. Ensure Delete Business Object button event handler calls DoDeleteAndClose
11. Ensure all changes to the body's business objects use the form's UnitOfWork (declared in the CRUDBusinessObjectForm base class).
12. Override ReloadModel to load the model with the business object's OID into the form's FormHeadBusinessObject
13. Override ModelToView to perform any actions required to sync the view from the model. This usually includes setting the Xpo.XPBindingSource's DataSource to the model business object.
14. Override ViewToModel to perform any actions required to sync the model from the view. For example, copy data that couldn't be bound using data binding from the view to the model. Or force close any open editors to ensure all changes are stored in the view.
15. Optionally override ReloadModelAndView. The default logic calls ReloadModel, then ModelToView.
16. Optionally override ProcessDataStoreException to map any commit errors to a user-friendly error message.

Review ViewRecipeCard then EditGroceryList then EditRecipeCard in that order as they get progressively more complex.

3.1.4.1.10 Maintain Proper View State - Design

Base class MDIChildForm provides a virtual function UpdateViewState. All forms override this function and set the view state based on the data in the form and current view status, e.g., disable "Delete Row" button if there isn't a selected or focused row in the grid. All forms add a call to DoUpdateViewState in code and event handlers wherever conditions could require a change in view state. For example, at the end of InitializeComponent and ModelToView routines. Review all calls to DoUpdateViewState in EditGroceryList and EditRecipeCard for examples.

3.1.4.1.11 Enforcing Business Rules - Design

The previous sections describe the design for CRUDing bodies of BusinessObjects. The following sections explain the design for ensuring data quality and enforcing business rules. Specifically - Data type constraints and business rules that can be enforced within the body of BusinessObjects are enforced on the client. All others are enforced by the DataStore.

Rules enforced on the client – Data Type or within a single body of BusinessObjects

Property is required validation and enforcement

Using XPO, attach a Nullable(false) annotation to a property. XPO declares the database column, not null. BusinessObject.GetPropertyRequiredErrors triggers off these annotations and automatically returns errors in GetPropertyError indicating the field is required on the form. As a result, no coding is necessary for "xxx is required" property validation. See how GetPropertyRequiredErrors generates - RecipeCard.Title and Ingredient.ItemDescription required - for examples.

String masking validation and enforcement

Mask validation: BusinessObject.GetPropertyMaskErrors checks the name of the property and validates mask errors based on its property name suffix.

If property ends with –

- "email" – validate email address in the property
- "cellnumber"or "homenumber" or "mobilenumber" – validate US phone number in the property
- "zipcode" – validate US zipcode number in the property

Mask enforcement in views:

MDIChildForm.SetMaskForAllTextEdits also checks the property names and sets TextEdit masks accordingly. All forms must call SetControlConstraints() at the end of their InitializeComponent2().

To add additional masks, update GetPropertyMaskErrors and SetMaskForAllTextEdits to include the new mask. See UserProfile and LoginForm for an example.

Maximum string length enforcement

Using XPO, attach a Size(<<maxLength>>) annotation to a property. XPO declares the database column as nvarchar(<<maxLength>>). See Ingredient.ItemDescription.

For all bound TextEdits on forms, MDIChildForm.SetControlConstraints walks all controls on the CRUD form, checks their binding properties and sets the TextEdit control MaxLength to the bound SIZE() annotation. All forms need to call SetControlConstraints() at the end of their InitializeComponent2().

For editable, bound grid columns, attach <<view>>.ShownEditor event handler to DevExpressUxUtils.gridView_ShownEditor_SetMaxLength in InitializeComponents2(). This looks at the column's bound property and sets the editor's MaxLength to the property's SIZE() annotation value. See EditRecipeCard's Ingredients grid and EditGroceryList's GroceryListItem grid for examples.

Rules enforced by the DataStore

There are common conditions that require DataStore validation and enforcement – Key uniqueness and required foreign key references.

Uniqueness validation and enforcement - Design

Some properties must be unique across other business objects; often business keys visible to the end user such as Account Number, Customer ID, etc. Enforce uniqueness by creating a database unique index or constraint. Use HeadBusinessObject. GenerateUniqueConstraintName to generate the name for the constraint. Create the constraint in the DataStoreService app server startup logic (specifically, its static CreateDataStore method). During SaveModel, if attempting to save a duplicate property, the DataStore raises the constraint which propagates to the client. CRUDBusinessObjectForm.ProcessDataStoreException catches the exception, extracts the property, matches it to the DataStore check constraint and sets the error message to the bound control on the form. See RecipeBoxSolutionModel.SQL.GenerateConstraintAndIndexesSQL() and ProcessDataStoreException to see how the system validates and enforces a unique RecipeCard.Title within each RecipeBox.

3.1.4.1.12 SeachRecipeBox Design

SearchRecipeBox uses XPO's XPInstantFeedbackView as the search results data source for scalability and performance. It only returns a page at a time to the client as the user scrolls, filters and sorts. As a result, filtering and sorting are performed by the DataStore. Note that RatingRatingCount (e.g., "4.70 (1201)") and TitleHyperLink properties are executed on the server to support XPO server mode views. See these properties in RecipeCard to see how this is done.

Step through the code

To see the design in action, set a breakpoint at the beginning of each of these methods and run the Simple search and edit RecipeCard scenario above (i.e., simple search, select, edit, change, save, change, saveandclose).

1. Program.Main
2. RecipeBoxMDIParent.RecipeMDIParent_Load
3. SearchRecipeBox.SearchRecipeBox_Load
4. filterSimpleButton_Click
5. editRecipeCardBarButtonItem_ItemClick
6. EditRecipeCard.EditRecipeCard_Load
7. barButtonSave_ItemClick
8. barButtonSaveAndClose_ItemClick
9. SimpleCRUDFramework.PushStackAndShowDialogForm executes when EditRecipeCard form closes which invokes the callback action in ShowEditRecipeCardForm. The action calls ShowNextForm(form)

3.1.4.1.13 Import AllRecipes.com Design

KIS: Fetch recipe web page from AllRecipies.com. Parse the recipe JSON embedded in the web page.

3.1.4.1.14 Email GroceryList Design

OAuth2 client to Gmail's email service. See Foundation.EmailUtils.

3.1.4.2 Security Design

Program.Main creates a XpoDefault.DataLayer over secure HTTPS WCF connection using credentials invoked from Windows Environment APIs (System Name, MAC Address) and email address.

Cache the email, encrypted in a file, one file for each Windows user account.

eRecipeBox client encrypts email, Windows System Name, Windows User Name and MAC Address into WCF user and pw, sends to UserNamePasswordValidator on server who performs validation. Rejects WCF connection if error.

Encrypt DataStore passwords, Azure & Openai API keys in the app.config.

User can only search their RecipeBox RecipeCards implemented by joining to RecipeBox when running filter.

3.1.4.3 Concurrency Design

Optimistic concurrency– Design

eRecipeBox supports multiple, simultaneous users using XPO's optimistic concurrency feature with its soft delete option. CRUD forms modify a body of BusinessObjects. CommitChanges() saves only the objects within the body of BusinessObjects that were modified. If another session simultaneously modifies any one of the same BusinessObjects in the UnitOfWork, an optimistic concurrency exception is thrown. As a result, different parts of body can be modified simultaneously by two different sessions (e.g., edit different Ingredients within a RecipeCard). XPO implements this by adding two columns to each table –

OptimisticLockField(int) - Increments a one up version for the row upon each save.

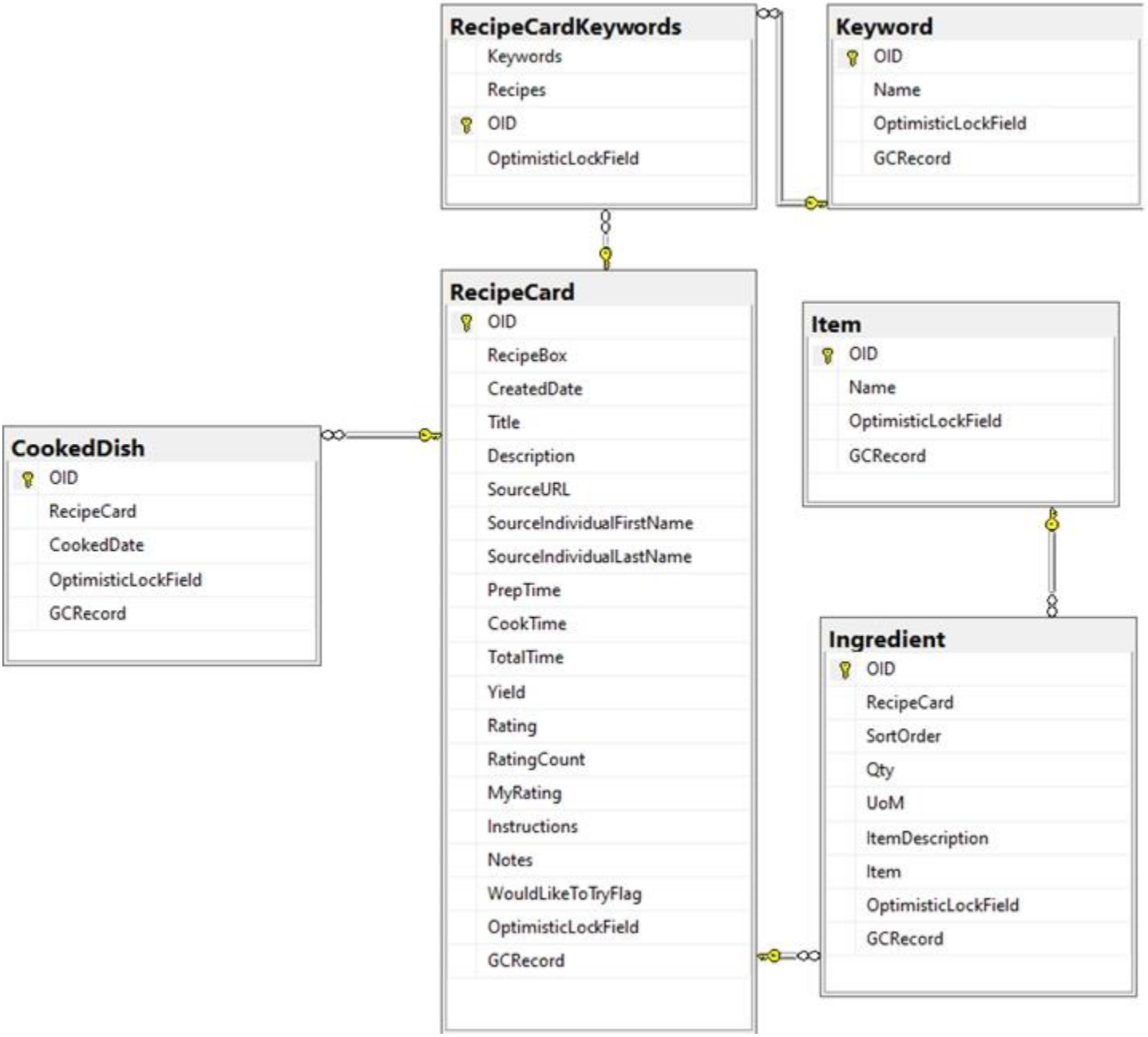
GCRecord(int) - 0 if live data. XPO assigns a unique int if the row is soft deleted. All foreign references to soft deleted rows are also set to null. GC operation deletes all rows where **GCRecord** is not null.

CRUDBusinessObjectForm. SaveModel catches the error and notifies the end user. Editxxx forms should use DoSave, DoSaveAndClose, DoDeleteAndClose and DoCancel methods. No other coding is required for new CRUD forms to support optimistic concurrency. See ViewRecipeCard, EditGroceryList and EditRecipeCard for examples.

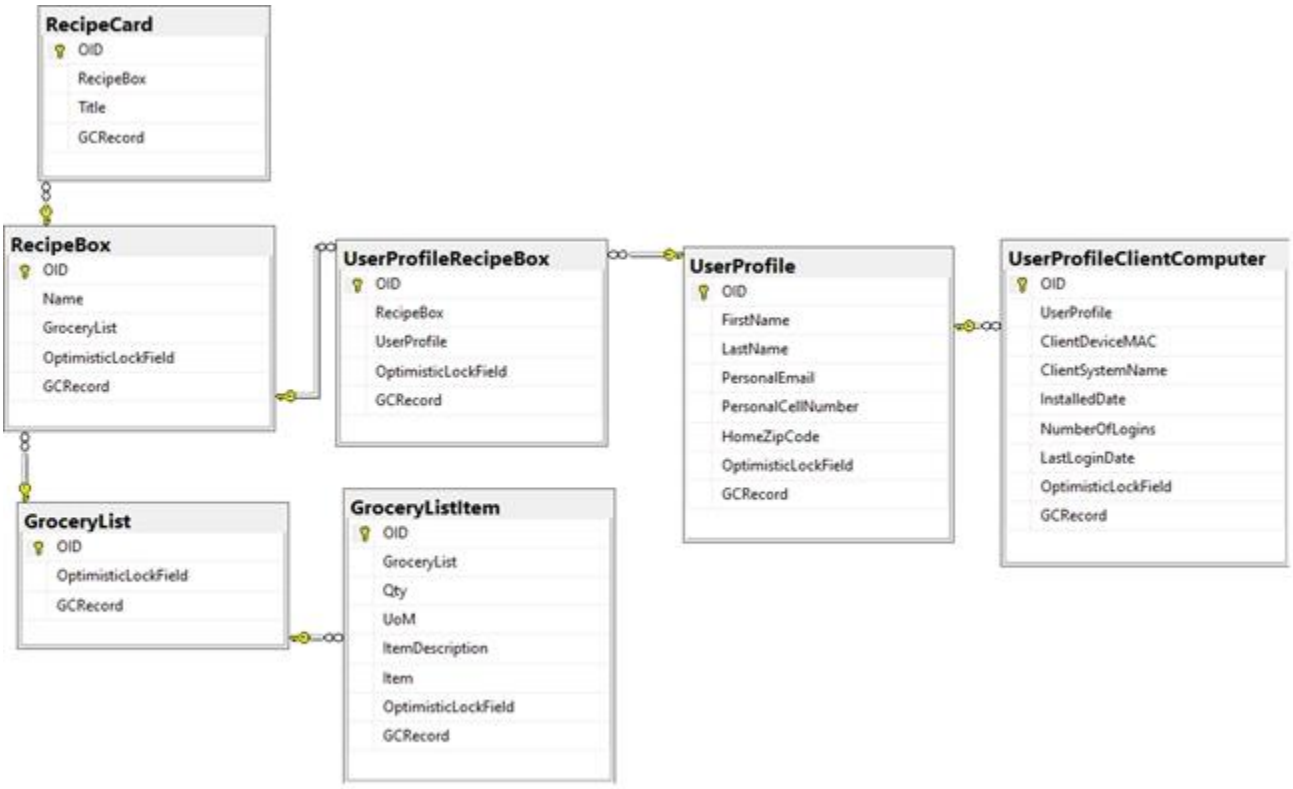
3.1.4.4 DataStore Design

XPO automatically creates the DataStore schema, driven by the BusinessObject classes and their annotations.

This E/R diagram shows the tables supporting the RecipeCard body of objects.



Remaining tables.



OptimisticLockField supports optimistic concurrency and is incremented with each save. GCRecord supports soft deletes. A unique number is set in GCRecord, indicating the soft delete, and XPO nulls any foreign key values in the same record. eRecipeBox close app event handler calls PurgeDeletedObjects(), which deletes all records with GCRecord with values.

3.1.4.4.1 eRecipeBox Data Dictionary

Table	Column	Data Type	Masking	Max
Cooked Dish	OID	int		4
Cooked Dish	Recipe Card	int		4
Cooked Dish	Cooked Date	datetime	date only	8
Cooked Dish	Optimistic Lock Field	int		4
Cooked Dish	GC Record	int		4
Grocery List	OID	int		4
Grocery List	Recipe Box	int		4
Grocery List	Optimistic Lock Field	int		4
Grocery List	GC Record	int		4
Grocery List Item	OID	int		4
Grocery List Item	Grocery List	int		4
Grocery List Item	Qty	nvarchar		510
Grocery List Item	UoM	nvarchar		510
Grocery List Item	Item Description	nvarchar		8000
Grocery List Item	Item	nvarchar		8000
Grocery List Item	Optimistic Lock Field	int		4
Grocery List Item	GC Record	int		4
Ingredient	OID	int		4
Ingredient	Recipe Card	int		4
Ingredient	Sort Order	int	0, 1, 2, etc.	4
Ingredient	Qty	nvarchar		510
Ingredient	UoM	nvarchar		510
Ingredient	Item Description	nvarchar		8000
Ingredient	Item	int		4
Ingredient	Optimistic Lock Field	int		4
Ingredient	GC Record	int		4
Item	OID	int		4
Item	Name	nvarchar		8000
Item	Optimistic Lock Field	int		4
Item	GC Record	int		4
Keyword	OID	int		4
Keyword	Name	nvarchar		8000
Keyword	Optimistic Lock Field	int		4
Keyword	GC Record	int		4
Meta Data Version	OID	int		4
Meta Data Version	DataStore Component	nvarchar	"SchemaDefinition" or "ReferenceData"	512
Meta Data Version	Version	datetime	ddmmyyyyhhmmss	8
Meta Data Version	Is Major Version	bit	0=false, 1=true	1
Meta Data Version	Comment	nvarchar		8000
Meta Data Version	Optimistic Lock Field	int		4
Meta Data Version	GC Record	int		4
Recipe Box	OID	int		4
Recipe Box	Name	nvarchar		8000
Recipe Box	Optimistic Lock Field	int		4
Recipe Box	GC Record	int		4
Recipe Card	OID	int		4
Recipe Card	Recipe Box	int		4
Recipe Card	Created Date	datetime		8
Recipe Card	Title	nvarchar		2000
Recipe Card	Description	nvarchar		8000
Recipe Card	Source URL	nvarchar	URI	2000

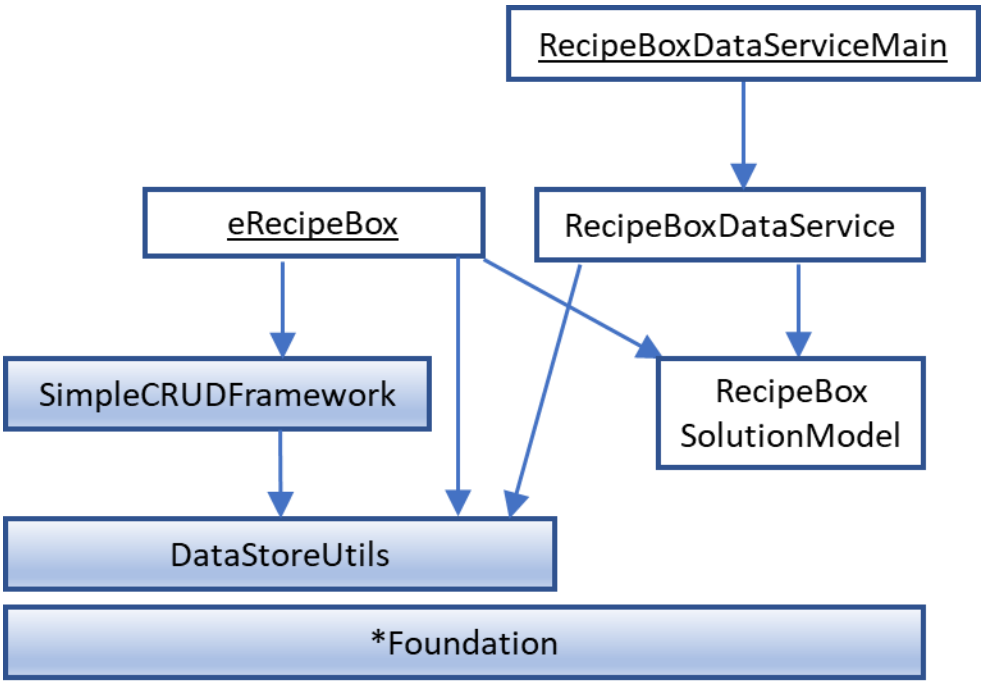
Recipe Card	Source Individual First Name	nvarchar		500
Recipe Card	Source Individual Last Name	nvarchar		500
Recipe Card	Prep Time	int	in minutes	4
Recipe Card	Cook Time	int	in minutes	4
Recipe Card	Total Time	int	in minutes	4
Recipe Card	Yield	nvarchar	in servings	400
Recipe Card	Rating	money	decimal w/ 2 precision	8
Recipe Card	Rating Count	int	0,1,…	4
Recipe Card	My Rating	int	decimal w/ 2 precision	4
Recipe Card	Instructions	nvarchar		8000
Recipe Card	Notes	nvarchar		8000
Recipe Card	Would Like To Try Flag	bit		1
Recipe Card	Optimistic Lock Field	int		4
Recipe Card	GC Record	int		4
Recipe Card Keywords	Keywords	int		4
Recipe Card Keywords	Recipes	int		4
Recipe Card Keywords	OID	int		4
Recipe Card Keywords	Optimistic Lock Field	int		4
User Profile	OID	int		4
User Profile	First Name	nvarchar		1000
User Profile	Last Name	nvarchar		1000
User Profile	Personal Email	nvarchar	RFC 5322 compliant	1000
User Profile	Personal Cell Number	nvarchar	USPhone NumberMask = @"\d{10}"	100
User Profile	Home Zip Code	nvarchar	USZip CodeMask = @"\d{5}"	20
User Profile	Optimistic Lock Field	int		4
User Profile Client Computer	OID	int		4
User Profile Client Computer	User Profile	int		4
User Profile Client Computer	Client Device MAC	nvarchar		200
User Profile Client Computer	Client System Name	nvarchar		512
User Profile Client Computer	Windows User Name	nvarchar		512
User Profile Client Computer	Installed Date	datetime		8
User Profile Client Computer	Number Of Logins	int	0,1,…	4
User Profile Client Computer	Last Login Date	datetime		8
User Profile Client Computer	Optimistic Lock Field	int		4
User Profile Client Computer	GC Record	int		4
User Profile Recipe Box	OID	int		4
User Profile Recipe Box	Recipe Box	int		4
User Profile Recipe Box	User Profile	int		4
User Profile Recipe Box	Optimistic Lock Field	int		4
User Profile Recipe Box	GC Record	int		4
XPObject Type	OID	int		4
XPObject Type	Type Name	nvarchar		508
XPObject Type	Assembly Name	nvarchar		508

3.1.4.5 Component Design

The Component Design describes how the implementation is packaged into components and shows their interdependencies. It also lists third party components and platforms.

3.1.4.5.1 eRecipeBox Components

Following are eRecipeBox's components and their package dependencies.



*Note: All components import **Foundation**

Shaded components are reusable components that support any CRUD application. They have no knowledge of eRecipeBox.

Foundation – Wraps O/S & environment APIs and provides utility functions on scalar data types.
DataStoreUtils – Utility functions on XPO objects and provides a datastore provider independent API for backup and restore. SQLite, SQL Server, and PostgreSQL providers are supported.
SimpleCRUDFramework – Provides framework for creating forms that CRUD XPO BusinessObjects, including common CRUD error messages. Also provides simple form navigation management (FormStack).
RecipeBoxSolutionModel – eRecipeBox ToBe Solution Model BusinessObjects implemented as XPO XPObjects.
eRecipeBox – Main WinForms application. All forms and importers.
RecipeBoxDataService – XPO WCF HTTPS DataStoreService supporting XPO client requests.
RecipeBoxDataServiceMain – Console app RecipeBoxDataService.

Note: DataStoreUtils can establish a connection to RecipeBoxDataServe or directly to a DataStore provider (SQL Server, Postgres, SQLite) through an app.config setting. This enables eRecipeBox client to deploy in 2 or 3-tier configurations.

3.1.4.5.2 SDKs & Third Party Components

eRecipeBox V1.0 is built using the following languages, SDKs and 3rd party components -

- C#
- Microsoft's .Net Framework

- Microsoft & DevExpress WinForms Components
- WCF over HTTPS for client/middle tier communication
- DevExpress XPO ORM supports providers - Advantage, ASA, ASE, DB2, Firebird, In-Memory, Microsoft, Microsoft, Microsoft Access, MySql, Oracle, Pervasive, PostgreSQL, SAP HANA, SQL Server, SQL Server Compact, SQLite, VistaDB
- eRecipeBox's supports the following DataStores -
    - SQLite – single user, development only
    - SQL Server
    - PostgreSQL

Top level Git package references for each component: Reference the latest eRecipeBox blueprints in the eRecipeBox Source Package for the latest configuration.

	RecipeBoxDataServiceMain
eRecipeBox • HtmlAgilityPack • DevExpress WinForms V2x.x.x • DevExpress.Xpo V2x.x.x	RecipeBoxDataService • DevExpress.Xpo V2x.x.x
SimpleCRUDFramework • Micosoft.CognitiveServices.Speech • DevExpress WinForms V2x.x.x • DevExpress.Xpo V2x.x.x	
DataStoreUtils • DevExpress.Xpo V2x.x.x • Npgsql 7.0.6 • System.Data.SQLite	RecipeBoxSolutionModel • DevExpress.Xpo V2x.x.x
Foundation • Castle.Core • MimeKit • NLog • Google.Apis.Gmail.v1 • Pluralize.NET • DevExpress.Xpo V2x.x.x	

Tricky: Must use Npgsql 7.0.6. Upgrading to version 8 fails. XPO doesn't support V8.

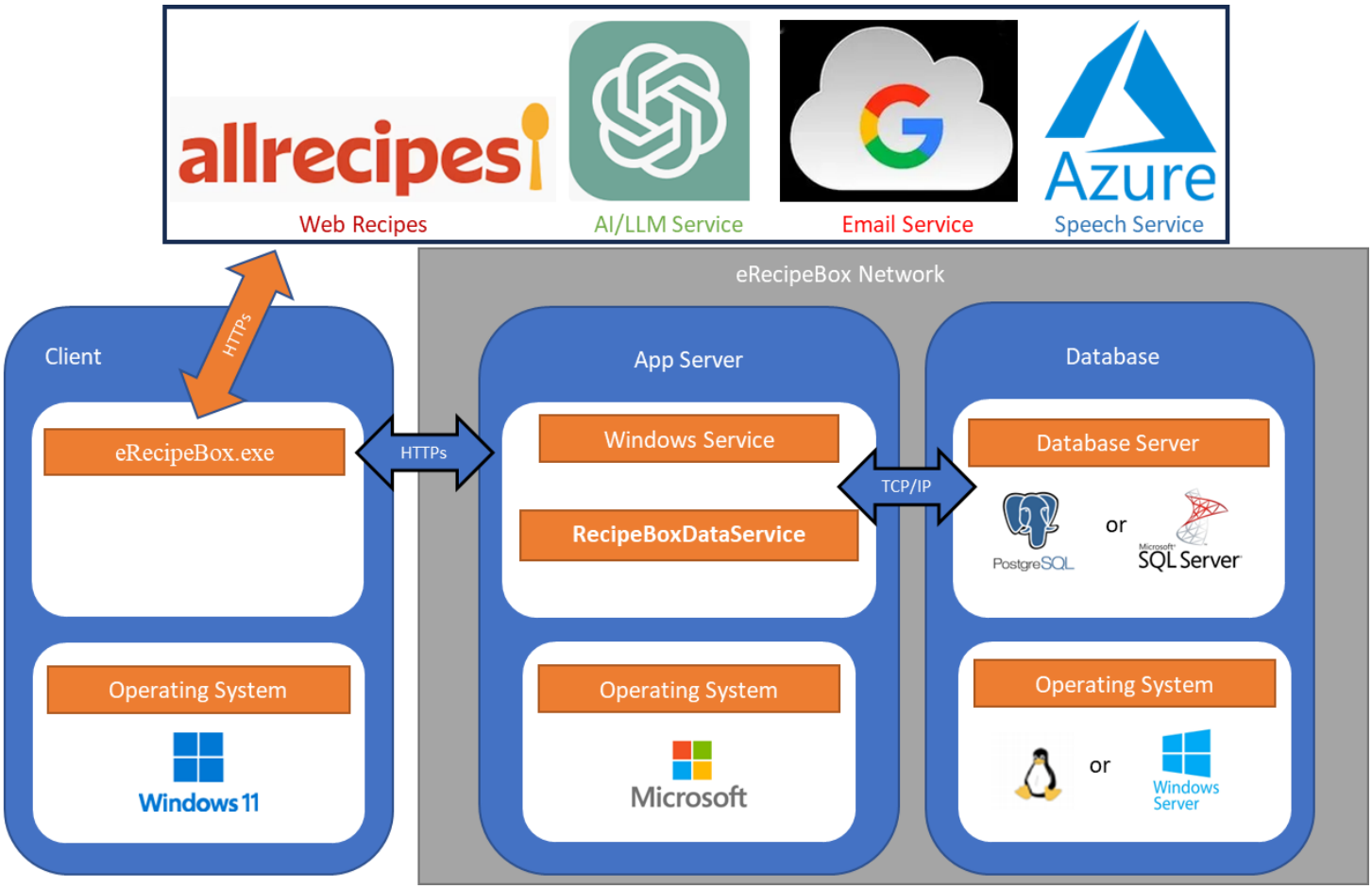
eRecipeBox client installer package: Inno Setup V6.2.2

3.1.4.6 Topology Design

The Topology Design communicates "the way in which constituent parts are interrelated or arranged." AwB captures Topology Design decisions in two views- Service Design and Network Design.

3.1.4.6.1Service Design

The Service Design diagram has become popular and is often presented as a System Design in many GitHub ReadMe files.



eRecipeBox Service Design

eRecipeBox client connects with RecipeBoxDataServer via WCF over https using: WCF user name and pw. Only accepts connection if passes authentication check.

==========

Commentary: This is a good example illustrating levels of detail. The Solution Model context shows eRecipeBox as a single black box with interfaces to Azure, Gmail, AllRecipes, and GPT. Because we need to deliver MVP 1.0 ASAP, we chose to call them from the client. Longer term Gmail and GPT interfaces definitely belong on the server (and perhaps also AllRecipes & Azure). Moving some or all of them to the app server is left as a student hands-on exercise. The Solution Model is not impacted. We only need to update the Topology Design to reflect our decision change.

==========

As noted in the Component Design, eRecipeBox client can be deployed in 2-tier configuration by modifying settings in the app.config. This makes development easier, and supports the SingleUser configuration (using SQLite). The projects deploys SingleUser versions to the business team after each Sprint so they can experience the app and provide more valuable feedback. See Production Environment for app.config settings.

3.1.4.6.2 Network Design

Network engineers rely on the Network Design to install and configure all nodes that host the System.

eRecipeBox Network Design

3-Tier Deployment

DataStore tier – Linux or Windows server hosting SQL Server or PostgreSQL behind a firewall who only allows the app server connection requests through.

Application Server tier – Windows app server hosting **RecipeBoxDataService** behind a firewall that redirects a port to it to service eRecipeBox requests.

Client tier – Windows 10 or 11 hosting **eRecipeBox** app supporting the core business use cases. Mobile email is used to view the GroceryList at the grocery store.

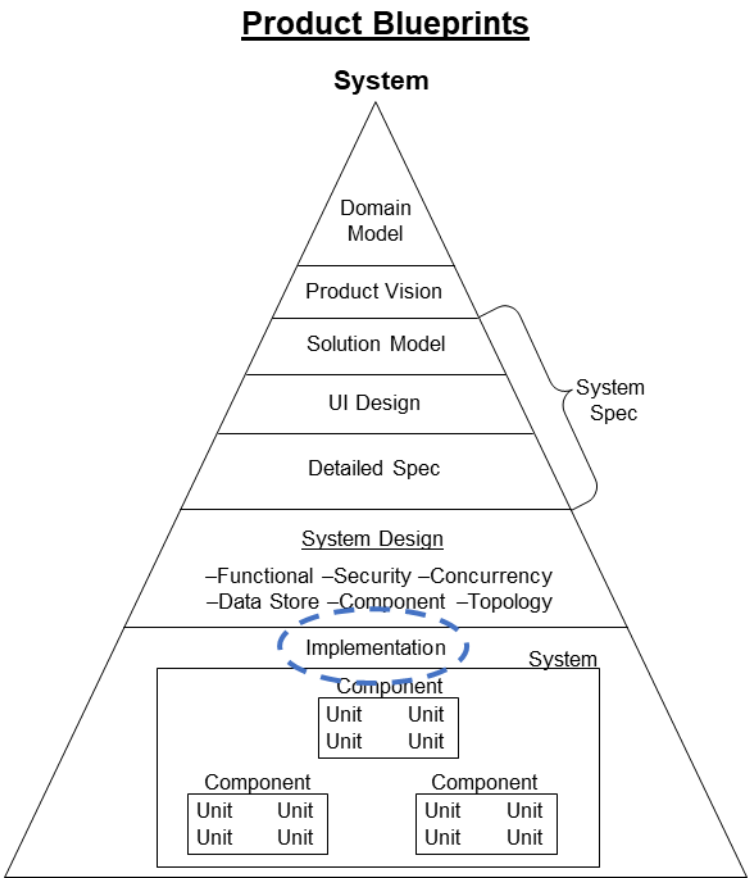
Note: The System can also be configured for 2-Tier and 1-Tier deployments through settings in eRecipeBox's configuration file.

==========

Commentary: Network Design is where network engineers specify load balancers, web server farms, caching servers, firewalls, reverse proxy servers, scalable storage devices, etc. Present the big picture network topology, then specify the HW and network configuration for each node.

==========

3.1.5 System Implementation

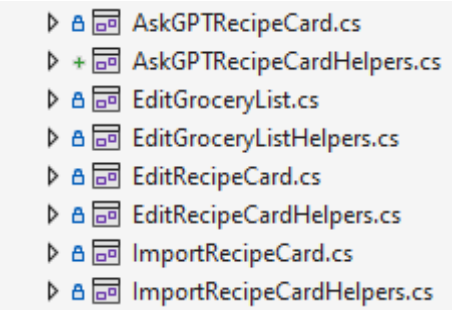


3.1.5.1 Coding Standards

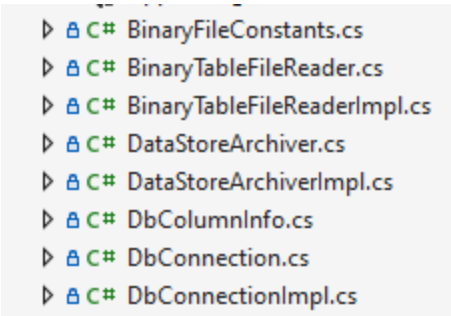
eRecipeBox follows Microsoft's coding conventions …

…with these additions -

Separate the public interface from the implementation into two source files for each unit. This makes it easy to first understand the purpose and behavior of a class before looking at its implementation. For WinForms classes, include only the constructor and Visual Studio generated event handlers in the interface file. Place all other methods in a file appended with "Helpers." e.g.,

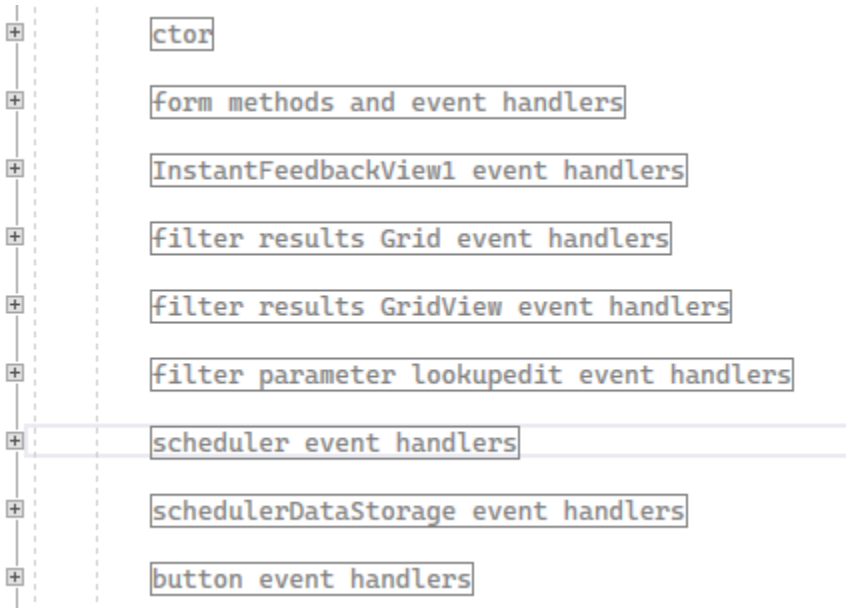


For non-WinForms classes, append the implementation file with "Impl." e.g.,



Append utility class names (that usually wrap another class and/or have statics) with Utils.

For WinForms classes, use #regions. Create a region to contain all event handlers for each significant control, followed by the button event handlers. See Dev Environment for instructions to install Visual Studio extension to collapse all regions. SearchRecipeBox example:



For WinForms classes, create and call `InitializeComponent2()` in the constructor to encapsulate eRecipeBox initialization logic. Implement `InitializeComponent2()` in xxHelpers file.

Use this qualifier order for classes and members

```
[static] [public|internal|protected|private] [delegate] [partial] [abstract|virtual|override]
```

Use descriptive names for identifiers. Descriptive identifiers communicate in calling code as well as declaring code. The goal is to minimize the need for comments, which tend to get out of sync over time. For example, this doesn't need a comment.

```
public override void EnableAllFKsAndRebuildIndexes()
```

Prefer explicit type declarations over "var"

```
No:        var selectedRowHandle = selectedRowHandles[i];
Preferred: int selectedRowHandle = selectedRowHandles[i];
```

Use these comment tags to annotate code.

#AtmtdTsting –Indicates this code is included to support automated testing.
#FRAGILE
#FRAGILE ASSUMPTION
#RQT [RqtID] <requirement description>
#TODO
#TODO BUG
#TODO PERFORMANCE
#TODO ENHANCEMENT
#TODO REFACTOR
#TRICKY
#WORKAROUND

3.1.5.2 Source Code

This video introduces the System Implementation structure. Projects mirror the components in the Component Design in Section 6.1.4.5.1. Extensive use of #regions keeps the code organized and manageable.

3.1.5.3 Version Control

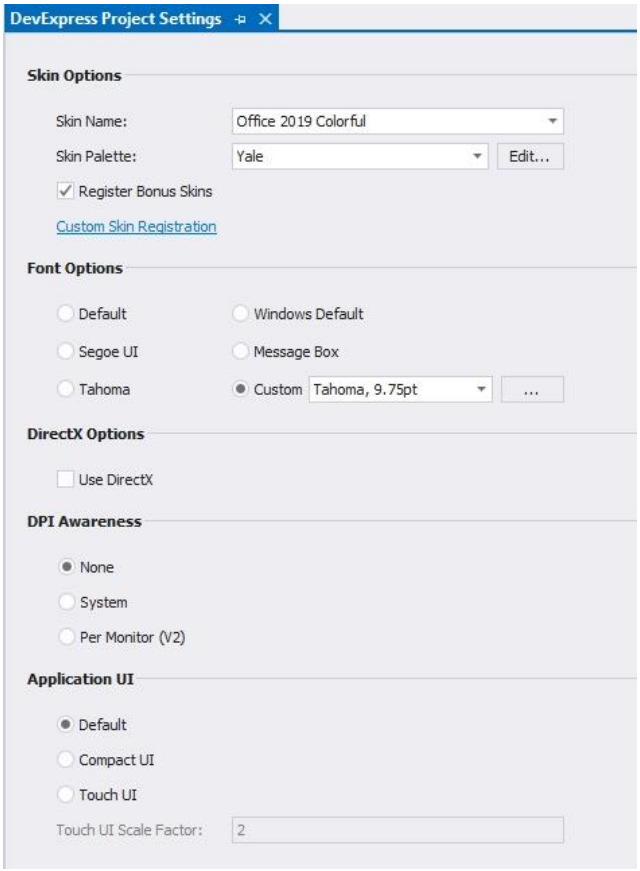
eRecipeBox source is managed in a GitHub repo. For clarity, the integration branch name corresponds to its DevExpress component version and append eRecipeBox version at the end of the tag, e.g., branch V23.3.6 contains builds against DevExpress version V23.3.6 with tags V23.3.6.1, V23.3.6.2, etc. When DevEx V24.1.3 is released, create branch V24.1.3 with tags V24.1.3.1, V24.1.3.2, etc.

eRecipeBox Source Package and a link to its GitHub repo can be found here.

3.1.5.4 Build the System

eRecipeBox client: *eRecipeBox.sln* –

DevExpress Project Settings: Disable DirectX, use windows Default Font, DPI awareness to None(optional), application UI = default. If not, will experience font issues on Windows 11 Surface.



eRecipeBox DataStore Service: *RecipeBoxDataServiceMain.sln*

Important! These must be back-reved (in 2-tier client or 3-tier app server app.config) in order for Npgsql 7.0.6 to work.
System.Text.Json 7.0.0.0
System.Diagnostics.DiagnosticSource 6.0.0.0
Microsoft.Extensions.Logging.Abstractions 6.0.0.0

To create **eRecipeBoxSetupV2x.xx.xx.exe** self-extracting installation script.

1. Clean out logs, etc.
2. Update app.config
3. Build eRecipeBox.sln client.
4. Update script ...\eRecipeBoxSystem\ eRecipeBox\ Installer\ eRecipeBoxInstallerScript.iss (app version and OutputDir).
5. Launch Inno Setup Compiler app.
6. Open the .iss script run it

==========

Commentary: For projects with larger teams, document the continuous integration (CI) process in this section, e.g., Commit code, push to remote repo, trigger CI pipeline (pull latest, run static code analysis, manage dependencies, build, run tests, run dynamic analysis and security scans, generate test & coverage reports, notify stakeholders, package & archive release, deploy release)

==========

3.1.5.5 System Implementation Metrics

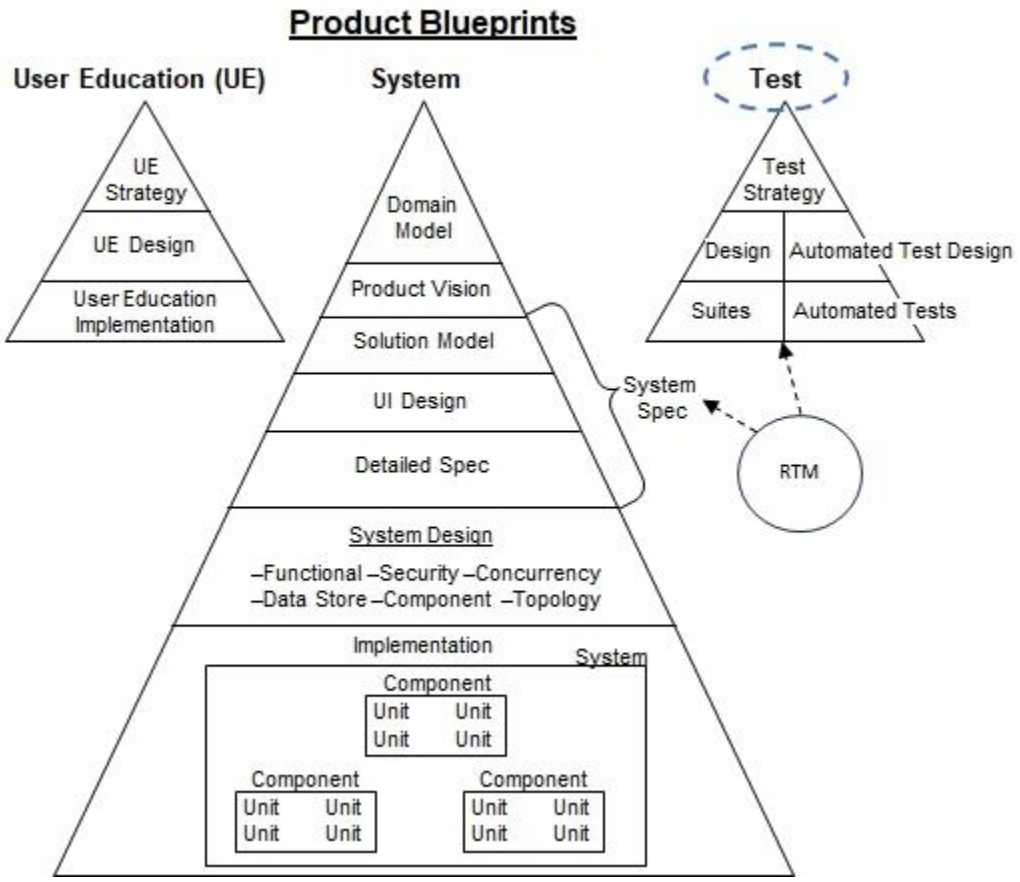
Summary metrics for the system implementation. Visual Studio - View.Other Windows.Code metrics for eRecipeBox.

eRecipeBox client and RecipeBoxDataService solutions

Project	Maint Idx	Cyclo Cmplxty	Depth of Inher	Class Cpling	SLOC
DataStoreUtils	85	442	2	129	2204
DocumentGenerator	67	97	3	69	856
eRecipeBox	68	888	13	447	10556
RecipeBoxDataService	69	17	3	22	231
RecipeBoxDataServiceMain	61	1	1	5	31
Foundation	72	261	6	127	1465
RecipeBoxSolutionModel	81	254	7	48	1703
SimpleCRUDFramework	79	291	12	176	1872

3.2 Test Blueprints

Agile w/ Blueprints Test artifacts capture system Test decisions in three levels of detail.



1. **Test Strategy:** Communicates the time and resources to be invested toward system testing and, using these constraints, describes the overall testing approach. This includes approaches for test data, test automation, and test support tools. It enumerates all System Test Suites and specifies their respective scope.
2. **Test Suite Design:** For each Test Suite identified in the Test Strategy, describe the paths and conditions that the tests will drive the system. It also summarizes test data required for the Test Suite.
3. **Test Suites:** Each Test Suite contains a set of Test Cases. Each Test Case narrates a sequence of Test Steps and their expected results. Test Cases can be manual or automated test scripts.

Many (if not most) business applications employ automated test scripts in order to reduce long-term maintenance costs. This automation needs to be designed and implemented in the same manner as the System itself.

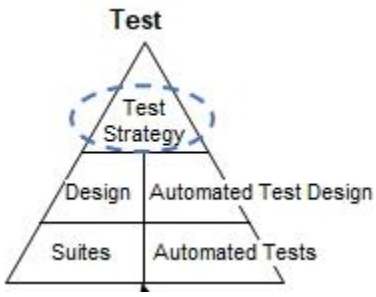
4. **Automated Test Design:** Describes the Test framework design using Agile w/ Blueprints' 6 view design, where appropriate.
5. **Automated Tests:** A set of automated Test scripts that exercise the System as specified in the Test Suite Design.

Last, ensuring test coverage with a Requirements Traceability Matrix is often beneficial, if not mandatory for regulatory environments.

6. **Requirements Traceability Matrix (RTM):** Reports the product requirement(s) verified by each system Test Case. The RTM can be generated as a report if each Test Case annotates its requirements being verified.

3.2.1 Test Strategy

eRecipeBox is a relatively small app with ~12 Solution Model classes and ~7 UI forms. The project invests four person days to design System Test Suites, Automated Test Design, and create system test data. The project also allocates 12-14 person days to implement Automated Tests and execute all System Tests.



3.2.1.1 System Test Approach

- Create a single test data set that covers all classes, properties, and associations using actual recipes from the internet, legacy recipe cards and cookbooks.
- All Test Suites use the same test data set.
- Leverage automated testing. This article is a good summary of the automated test tool options for eRecipeBox. What are the best UI Test Automation Tools? (microsoft.com)
- Microsoft recommends Appium with WinAppDriver for testing desktop and UWP apps. WinAppDriver and Desktop UI Test Automation (microsoft.com) So eRecipeBox selects this approach.
- Use Windows' accessibility tool, Inspect.exe, to view Windows element names when creating automated test scripts. Accessibility tools - Inspect - Win32 apps | Microsoft Learn
- Log all CRUD operations and queries to the DataStore API as JSON. Compare the execution log with the expected results log. This minimizes Assert checks in the test scripts and is easier to maintain.

3.2.1.2 System Test Suites

Test eRecipeBox using three Test Suites -

1. **Business Test Suite** – Validate all To-Be Solution Model use cases. Exercise eRecipeBox as it would be exercised in a typical operational scenario by a business user.
2. **Functional Test Suite** – Verify each UI form and Security Requirements comprehensively. Test every feature/button on each UI form. Test all data properties against their min/max values. Force all possible error conditions.

3. **Non-functional Test Suite** – Verify Non-Functional Requirements in the Detailed Spec, emphasizing concurrency and scalability requirements.

Taking risks into account, the following System Tests have the highest priority -

- Business Tests Suite – Verify the Solution Model use cases
- Non-functional Test Suite – Verify support for multiple concurrent users and verify data integrity.
- Given the very limited test time and resources, the comprehensive Functional Test Suite is assigned lowest priority. Prioritize the most complex UI forms and features.

3.2.1.3 System Test Requirements Levied on the System

The Test Strategy requires the following from the Development team -

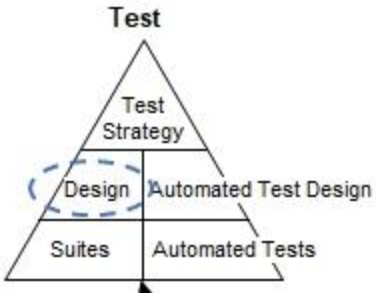
1. Implement ImportRecipeCard form in an early sprint to avoid having to manually write import RecipeCard tests scripts at the DataStore level in order to create a System Test dataset. Development also needs test data and will benefit from having Import functionality early on.
2. System Test needs to –
    a. Start Test Suite execution with a known DataStore state
    b. Execute tests (which modify the state) and
    c. Verify results against expected data.

    Tests Cases must be deterministic and repeatable, so the Test team needs an efficient way to back up the DataStore and restore the DataStore from backup. Therefore, System Test needs automated backup & restore (Requirement 6.1.3.3.1.8) to be implemented ASAP. See DataStoreUtils.DataStoreArchiver.
3. System Test needs a realistic, very large test data set for stress testing and capacity testing. Provide a way to generate large test data sets. See DataStoreUtils.DataStoreArchiver.ReplicateBusObjBodies.
4. Since System Test uses WinAppDriver for automated testing, Development must ensure every Windows Element on every form is assigned a unique name. Report these names to the Test team so they can be referenced in automated test scripts. See MDIChildForm.EndFormLoad and Log. ReportFormsAndTheirElements.
5. eRecipeBox opens the calendar to "today", so testing meal planning must use relative dates vs. absolute dates. Provide a way to move CookedDish dates to be the same relatively. See CookedDish.SetAnchorDate and AutomatedTestUtils.VerifySystemTestData.
6. eRecipeBox must log all calls to the DataStore - search and CRUD. System test scripts run a full test, then compare the DataStore log for the run with an expected log to ensure all DataStore interactions are the same. This dramatically reduces the amount of Asserts required in the test scripts themselves. See DataStoreUtils. XpoService.[InitiateQuery, LoadHeadBusObjByKey, CommitBodyOfBusObjs].
7. Provide a way for test scripts to set breadcrumbs in the DataStore log. This makes it easier to match test script location with the DataStore log. See Log. DataStore. Info($"Breadcrumb: {logBreadcrumbID}");

3.2.2 Test Suite Design

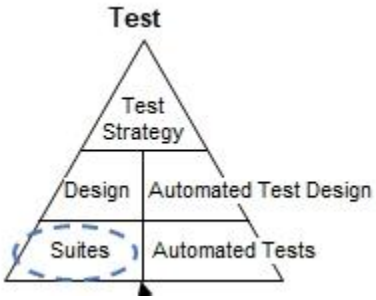
Following are the design scenarios for each test suite. Each Test Case is designed in two phases –

- Phase 1: The **Test Architect** creates the **Test Scenario Design**, which describes the high-level business scenario devoid of specific test data, i.e., narrate how the Test Case drives the app.
- Phase 2: The **Test Developer** creates the **App Scenario Steps** that reference specific test data.



Note: See eRecipeBoxTestSuiteDesign.xlsx for Test Suite Designs.

3.2.3 Test Suite Implementation



The following summarizes eRecipeBox System Test Suites.

Test Suite	Test Case	Auto Type	Suite Design	Suite Implementation
Business Test Suite	Business 01	Auto	See section 6.2.2.1	See section 6.2.5.
Functional Test Suite	Search Recipe Box 01	Auto	See section 6.2.2.2.1.	See section 6.2.5.
Functional Test Suite	Search Recipe Box 02	Man	See section 6.2.2.2.1.	Left as a student exercise
Functional Test Suite	Edit Recipe Card 01	Auto	See section 6.2.2.2.2.	Left as a student exercise
Functional Test Suite	Security 01	Man	See section 6.2.2.2.3.	Left as a student exercise
Non-Functional Test Suite	Non Functional 01	Man	See section 6.2.2.3.	Left as a student exercise
Non-Functional Test Suite	Stress Test 01	Auto	See section 6.2.2.3.	See section 6.2.5.

3.2.3.1 System Test Data

Business and Functional Test Suites use SysTest01 dataset.

Non-Functional Test Suite tests use 25KRecBoxes (SysTest01 replicated to 25,000 RecipeBoxes)

==========

DISCLAIMER:

This sample business app, including its accompanying test dataset, is provided for educational and learning purposes only. Users are encouraged to explore the features of the app and experiment with the test data as part of their leaning process

However, redistribution or sharing of the test dataset outside the context of this sample app is strictly prohibited. The test dataset included with this app is intended solely for the purpose of testing and learning within the scope of this sample application.

By using this sample app and its test dataset, you agree to abide by these terms and conditions. Any unauthorized distribution or sharing of the test dataset may result in legal action.

==========

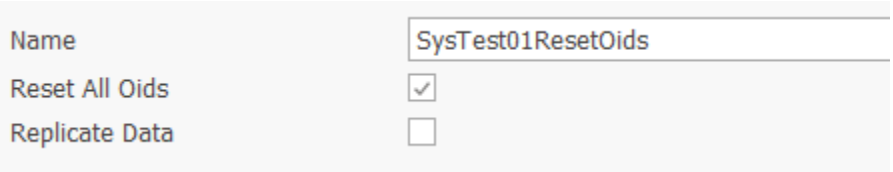
3.2.3.1.1 Large System Test Data

Security and Non-Functional Test Suites require a large DataStore containing 25,000 RecipeBoxes (SysTest01-25KRecBoxes).

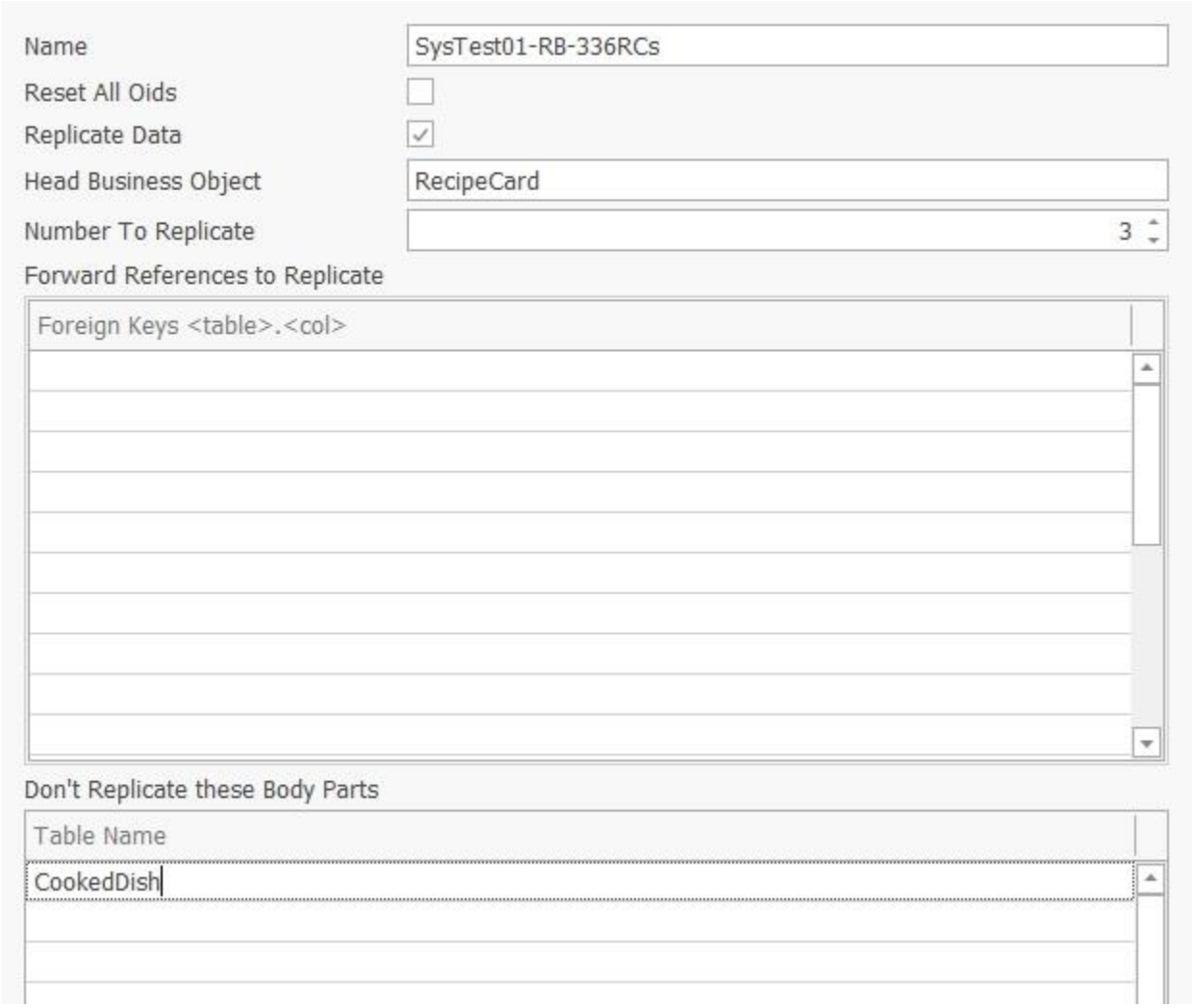
Steps to create SysTest01-25KRecBoxes.

Steps for creating a system test DataStore with 25,000 UserProfiles, each with their separate RecipeBox.

1. All Oids must be reset (eg, 1,2,3, …) in all tables in order for data replication to work. So, backup SysTest01 and reset its Oids.



2. Restore the SysTest01ResetOids snapshot.
3. SysTest01ResetOids has 1 UserProfile and 1 RecipeBox containing 85 RecipeCards. Make 3 copies of each RC in the RB so the RB contains 4*84 = 336 RCs. Do not replicate the CookedDishes though. We don't want to cook 4 copies of the RC on the same day.



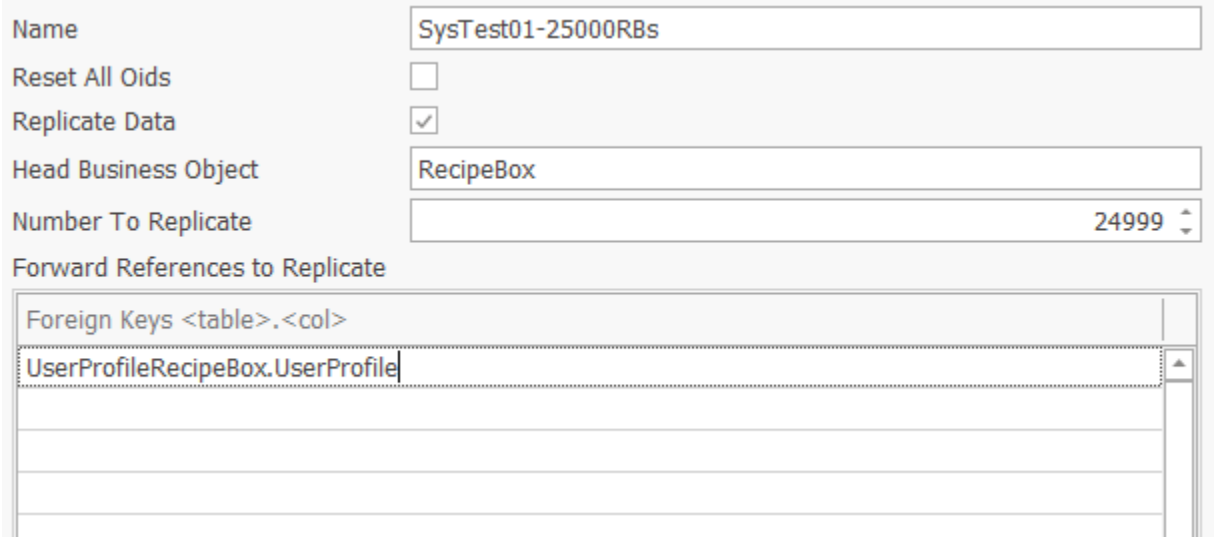
4. Restore the SysTest01-1RB-336RCs snapshot.

The replicator appends the Title with numbers in order to maintain its unique constraint. Example -

Rows: 336 Filter: <none>

Title	TotalTime (min)	Rating (Count)
Low Fat Cheesy Spinach and Eggplant Lasagna	85	4.77 (22)
Low Fat Cheesy Spinach and Eggplant Lasagna0002	85	4.77 (22)
Low Fat Cheesy Spinach and Eggplant Lasagna0003	85	4.77 (22)
Low Fat Cheesy Spinach and Eggplant Lasagna0004	85	4.77 (22)
Make-Ahead Vegetarian Moroccan Stew	70	4.59 (345)
Make-Ahead Vegetarian Moroccan Stew0002	70	4.59 (345)
Make-Ahead Vegetarian Moroccan Stew0003	70	4.59 (345)
Make-Ahead Vegetarian Moroccan Stew0004	70	4.59 (345)

5. DataStore now has 1 UserProfile and 1 RecipeBox containing 336 RecipeCards. Make 24,999 copies of the RecipeBox (and its contents which include GroceryList and RecipeCards). We want each replicated RecipeBox to get its own UserProfile, so replicate the UserProfileRecipeBox.UserProfile forward reference. Takes several minutes.



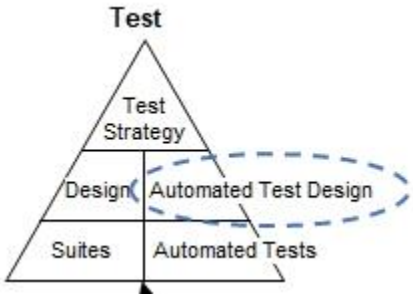
6. Restore SysTest01-25000RBs. Takes several minutes. 25K RecipeBoxes and UserProfiles. Replicates about 100M Ingredients.

Table Name	# Records
dbo.CookedDish	3,300,000
dbo.GroceryList	25,000
dbo.GroceryListItem	825,000
dbo.Ingredient	97,600,000
dbo.Item	295
dbo.Keyword	47
dbo.MetaDataVersion	8
dbo.RecipeBox	25,000
dbo.RecipeCard	8,400,000
dbo.RecipeCardKeywords	20,200,000
dbo.UserProfile	25,000
dbo.UserProfileClientComputer	25,000
dbo.UserProfileRecipeBox	25,000
dbo.XPObjectType	20

3.2.4 Automated Test Design

eRecipeBox Test Automation is built using the OpenQA.Selenium.Appium.Windows SDK. Following are the **Functional**, **Component** and **Topology** designs of the Automated Test Design.

This DevExpress article is a helpful resource that describes UI automation using Appium.



3.2.4.1 Automated System Test Functional Design

Dynamic Behavior- Typical Use Case

eRecipeBox Test Automation follows a typical Appium/WinAppDriver flow –

1. Create a Windows session by starting the Windows application.
2. Find a Windows UI Element
3. Click or SendKeys to the Element
4. Repeat steps 2 and 3
5. When finished, tear down the session
6. Compare the test run log containing DataStore action with test reference log containing expected DataStore actions.

Assigning IDs to Windows UI elements.

Development assigns IDs to each form and control in the app. If SystemTestLogging is true in eRecipeBox app.config, eRecipeBox logs all element IDs on all forms in \LogFiles\WindowElementNames.txt and FormEnums.txt during execution. See `MDIChildForm.SetAllControlNames`. Copy `enums` in FormEnums.txt into `eRecipeBoxSystemTests.eRecipeBoxWindowElements`.

Logging DataStore API Data and Using it to Validate Results

Automated testing is sensitive to the test data in the DataStore. Rather than issuing a query command in the UI and validating the expected results in the test script (usually with an Assert.IsEqual(UIElement.Text, "<expected result>"), eRecipeBox validates expected results by –

1. Logging all calls to the DataStore (`DataStoreUtils.XpoService)` during execution in \LogFiles\ DataStoreYYYY-MM-DD_hhmmss.log. Calls are logged in JSON format.
2. At the end of the test script, compare the eRecipeBox execution DataStore log with the expected results DataStore log. (See *BusinessTestSuiteExpectedDataResults.txt* as an example)

In order for this design not to be sensitive to internal DataStore data keys (e.g., OIDs and foreign keys), the JSON only includes business keys and business properties of business objects.

Automated Test Framework - Class Model

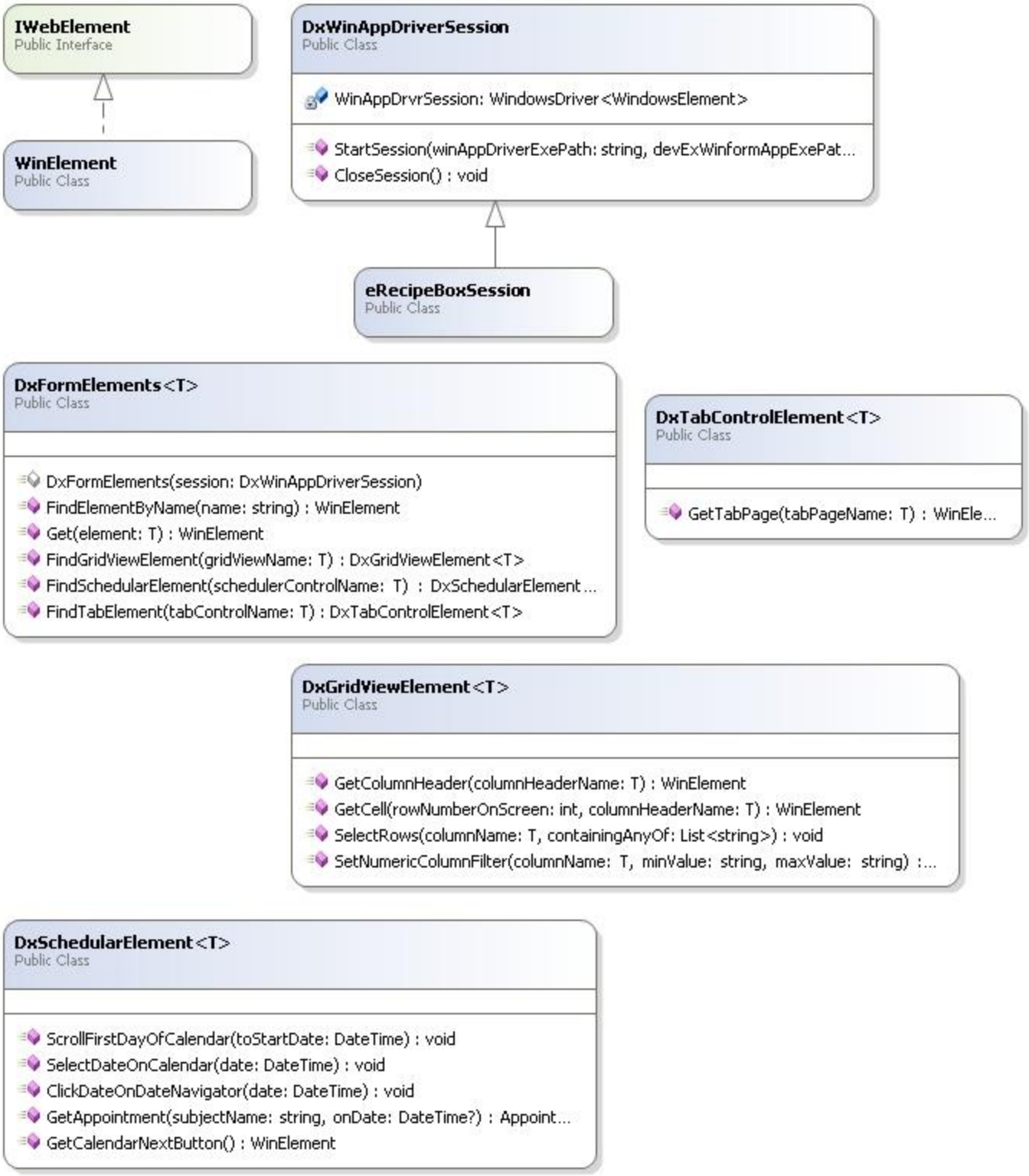
DevExpress does not formally support WinAppDriver for their WinForms product[i], so we wrote our own wrappers for their UI components used in eRecipeBox (by reverse engineering DevExpress' UI Element naming convention).

WinElement – Simple wrapper for Appium's IWebElement. Collects Click() response times.

DxWinAppDriverSession – Simple wrapper for a WindowsDriver<WindowsElement>.

DxFormElements<T> - Encapsulates the WinElements contained on a DevExpress RibbonForm. Provides methods to find both basic WinElements and complex elements on the form (specifically - gridView, schedular and tab)

There are also wrappers for each DevExpress component – XtraTabControl, XtraGrid and XtraSchedular.



Putting it all together

Using this design, automated test scripts typically follow this coding pattern
[Note: SRBs is a shortened name for SearchRecipeBox's elementIDs]

```
//Get Search form from a DxWinAppDriverSession
DxFormElements<SRBs> srb = new DxFormElements<SRBs>(TestSession);

//Get button from Search form and click it
srb.Get(SRBs.MonthView).Click();
```

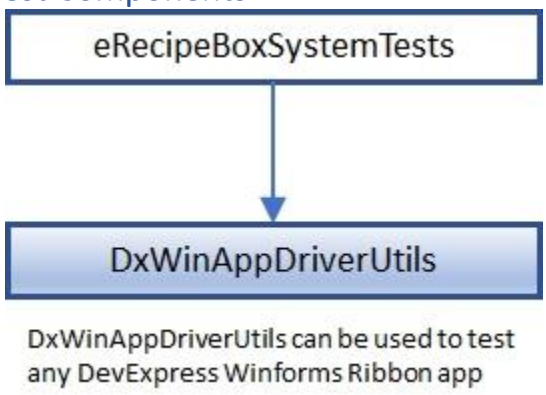
```
//Get GridView from Search form
DxGridViewElement<SRBs> searchResults = srb.FindGridViewElement(SRBs.searchResultsGridView);

//Get cell from GridView and click it.
WinElement row1Title = searchResults.GetCell(1, SRBs.Title);
row1Title.Click();
```

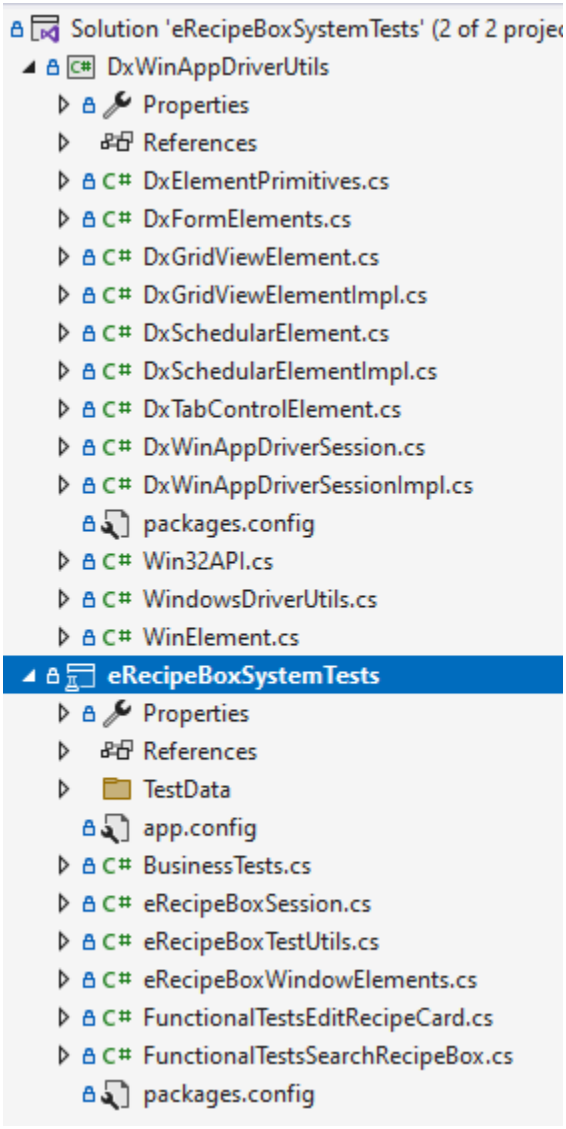
3.2.4.2 Automated System Test Component Design

This section describes the Automated Test framework components developed by the project team and the Automated Test third party components.

3.2.4.2.1 Automated System Test Components



The solution structure reflects the component design.



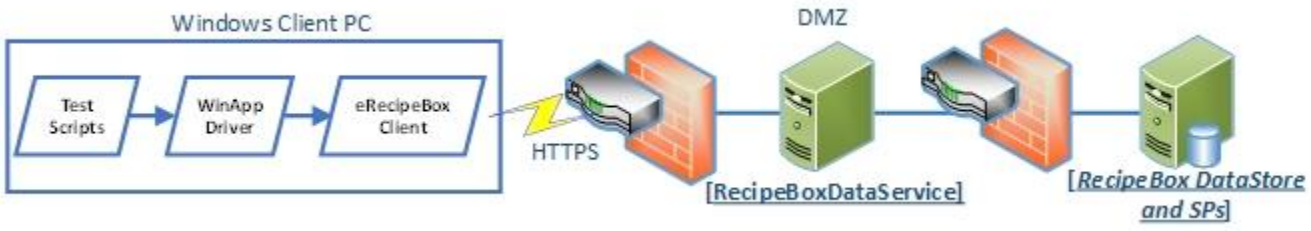
3.2.4.2.2 SDKs & 3rd Party Components

eRecipeBox Automated System Tests are built using the following languages, SDKs and 3rd party components:

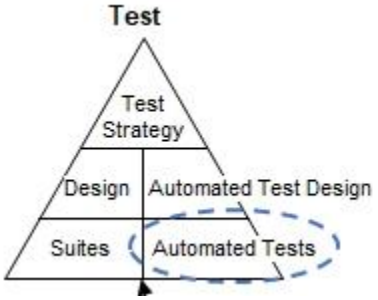
- C#
- Microsoft's .Net Framework
- Appium.WebDriver
- MSTest.TestFramework & MSTest.TestAdapter – MS Unit Testing Framework

eRecipeBox Automated Tests communicate with WinAppDriver over a local TCP socket.

3.2.4.3 Automated System Test Topology Design



3.2.5 Automated Tests Implementation



3.2.5.1 Automated Test Coding Standards

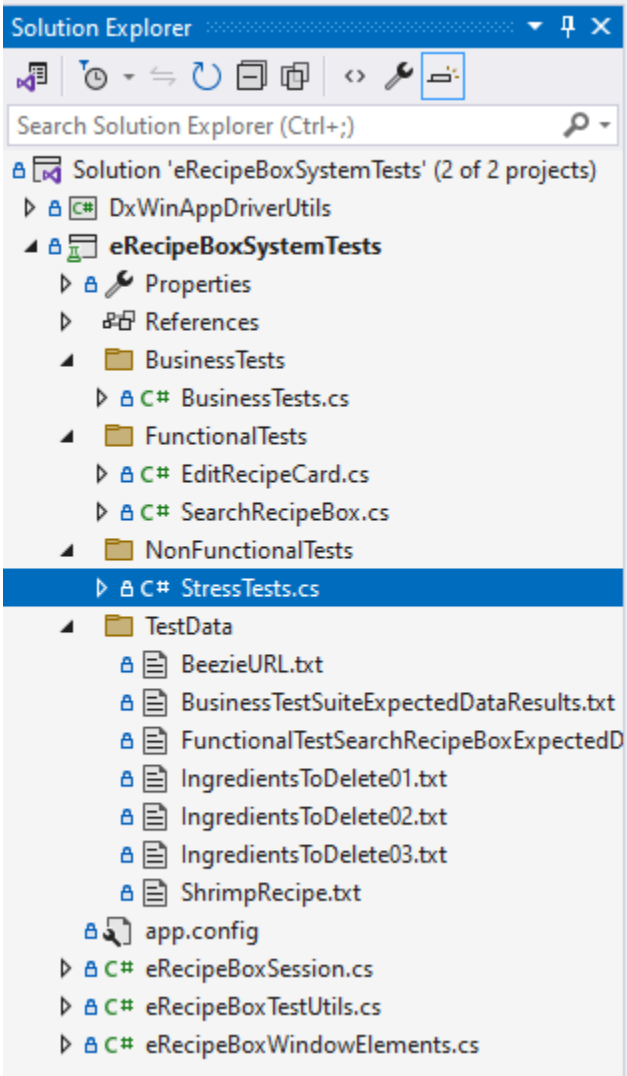
Follow same standards as System coding standards.

3.2.5.2 Automated Test Source Code

eRecipeBox MVP V1.0 contains the following automated system tests -

- BusinessTests.TestCase01
- FunctionalTests.SearchRecipeBox.TestCase01
- FunctionalTests.EditRecipeCard.TestCase01
- NonFunctionalTests.StressTests.TestCase01

Test Suite automated tests are located in eRecipeBoxSystemTests project.



3.2.5.3 Automated Test Version Control

eRecipeBox Test follows the same version control process as eRecipeBox System source.

3.2.5.4 Build System Tests

eRecipeBoxSystemTests.sln contains all automated System Tests.

3.2.5.5 Automated Test Executions

Test executions are typically stored as reports on a server or attached to a Sprint Task. For documentation purposes, we include the MVP 1.0 Test Execution results below.

3.2.5.5.1 MVP 1.0 Test Execution

This video demonstrates BusinessTests.TestCase01 execution.

This video demonstrates FunctionalTests.SearchRecipeBox.TestCase01 execution.

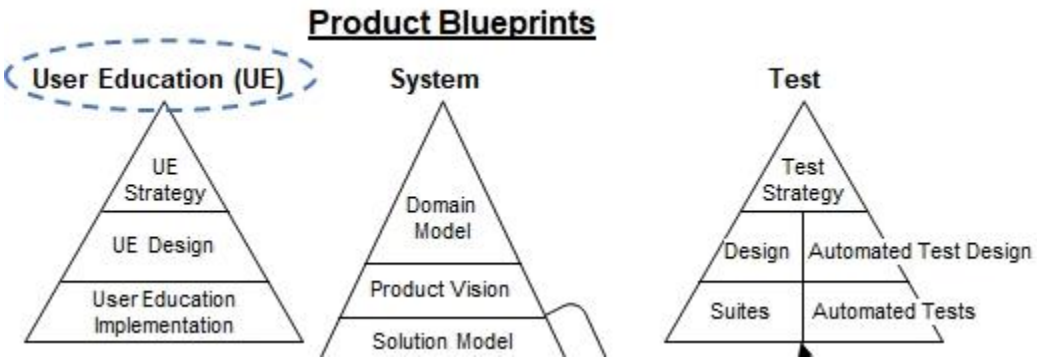
This video demonstrates NonFunctionalTests.StressTests.TestCase01 execution.

3.2.6 Requirement Traceability Matrix (RTM)

The RTM reports the set of requirements verified by each Test Case and vice versa. It highlights requirements that may not be fully tested. It also is helpful to determine which Test Cases need to be re-executed when fixing bugs and adding enhancements to the System.

The Test team didn't generate an RTM for eRecipeBox MVP 1.0 due to the aggressive development schedule. However, the Functional Test Suite Design traced requirements with each design step, so a partial RTM could be generated with a little effort. See eRecipeBoxTestSuiteDesign.xlsx for examples.

3.3 User Education Blueprints



3.3.1 User Education Strategy

Since the eRecipeBox UI follows Microsoft Office look and feel, and we assume our users are proficient with Microsoft Office, we minimize investment in user education. Create a short video demonstrating the core solution model use case and provide a link to it from help within the app. Also include tooltips for key shortcuts. Invest ½ day developing user education.

3.3.2 User Education Design

Create a video that walks thru typical usage scenario.

Schedule Meals for Next Week Use Case

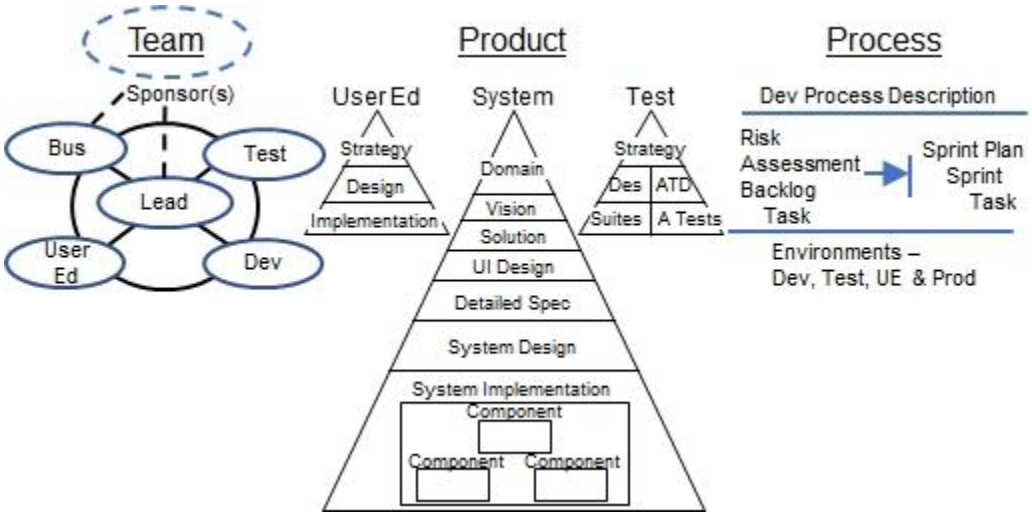
1. Search casseroles
2. View one of the resulting RecipeCards
3. Schedule it for next Monday
4. Search mediterranean and zucchini
5. Don't see anything user wants in results

6. Search AllRecipes.com and find a recipe
7. Import it
8. Schedule it for next Tuesday
9. Search vegan
10. View one of the resulting RecipeCards
11. Schedule it for next Wednesday
12. Realize family is going out on Monday, so move all 3 up one day
13. Import from GPT, asking for a dish that complements Thursday's meal
14. Schedule it for next Thursday
15. Generate a GroceryList for these meals
16. Add a few staples to the GroceryList
17. Email the GroceryList

3.3.3 User Education Implementation

This video is played when end user clicks Help.

4 Agile w/ Blueprints Team

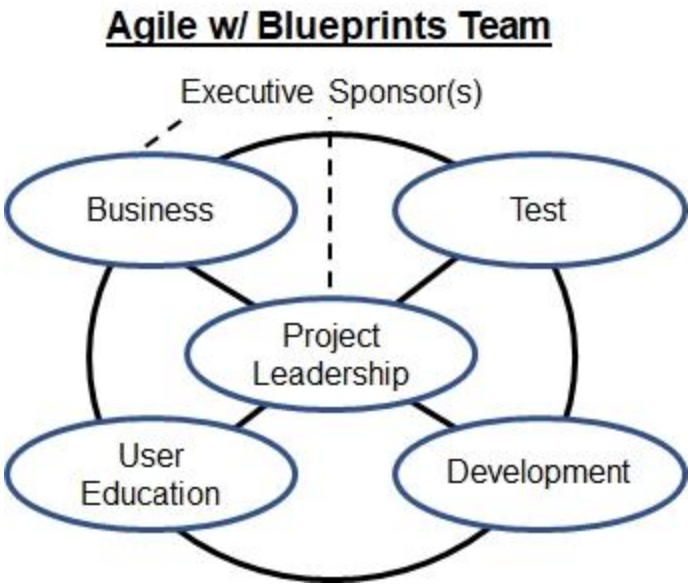


4.1 AwB Project Sub-Teams

Agile w/ Blueprints' project team is comprised of five sub-teams.

- **Business** – Business unit representatives participate in the requirements capture process, provide feedback, and prioritize features throughout the project. They ultimately approve System acceptance.
- **Project Leadership** – Fills the project management and requirements analyst roles.
- **Development** – Designs and implements the System.
- **Test** – Verifies the System implements the System Spec.
- **User Education** – Designs and implements User Education materials.

In addition, one or more executives provide strategic direction to the team and provide the necessary resources to build the product.



The project Team Document lists the roles within each sub team and assigns individuals to each of these roles.

Role	Description
Project Sponsors	
Business Executive Sponsor	The business executive sponsor identifies and prioritizes projects that support growth and/or improve the effectiveness of the business unit. Working with the IT Exec, the business exec allocates the resources required to develop the product.
IT Executive	The CIO/CTO typically has a significant role in defining the guidelines for technology, processes & tools for the project, ensuring they are consistent with corporate strategy. They also have a major role in allocating resources to the project.
Project Champion	The project champion's primary role is to raise & maintain visibility of the importance of the project and ensure the company makes the project a high priority. Occasionally, the business exec sponsor fills this role. However, in larger organizations, the project champion is more likely the business unit manager who most benefits from the product.
Business	
Business Unit Manager	Business unit manager(s) who most benefit from the project. They typically manage the majority of the end users of the product.
Lead Business User Representative	The Lead User Rep serves as the project point person for the entire business unit, providing two-way communication with Project Leadership. Typically, senior supervisors and/or business unit trainers who understand the current business processes and software tools the most. The Lead User Rep provides input to Project Leadership to develop the System Spec (blueprints) and is ultimately responsible for ensuring the Product meets the business needs. The Lead User Rep provides feature & schedule priorities when adding Backlog items to the Sprint Plan. Experience has shown a strong Lead User Rep role is critical to success.
Business User Representative	User Reps provide requirement and priority input for their area of expertise to the Lead User Rep and provide feedback on the product as it evolves throughout the Sprints.
Subject Matter Expert (SME)	SMEs provide targeted and focused input on highly specialized areas relating to the product. For example, when building a clinical medical application, MDs would provide SME input to the medical validity of the data or workflow of the system.
Project Leadership	

Team Lead	Team Lead grooms the Product Backlog, maintains the Risk Assessment & Sprint Plan, runs the daily CheckIn meetings & weekly Sprint Planning meetings, and Sprint Reviews. Most importantly, the Team Lead continuously finds creative solutions to overcome blockers.
Product Owner	Drives the market vision for the Product. Advocate for the business unit and business users. Prioritizes Product features and serves as the point between Project Leadership and Business teams. Ultimately responsible the market success of the for product. The product is their baby.
System Analyst	Responsible for absorbing & mastering the business domain and its challenges. Crafts a business Solution that can be built within project constraints.
UI Architect	Responsible for designing the UI conceptual framework and UI design.
Development	
System Architect	Responsible for the software architecture and design of the System.
Data Architect	Responsible for the persistent data model design.
Developer	Developers code and unit test the System implementation.
Graphic Designer	Responsible for creating any custom icons, images, and other graphical elements required by the UI. Selects the color scheme.
System Integrator	Maintains Environments documentation, responsible for developing and maintaining the tools that pull from the product repository, integrate, & build the System. Also responsible for developing tools that deploy and configure the System into their various environments.
Test	
Test Architect	Responsible for creating the System Test Strategy and designing the Test Suites. Also is responsible for designing the Automated Testing framework.
Test Developer	Creates the manual tests and automated test scripts. Also, creates supporting test data.
System Tester	Executes System Tests and creates issue Tasks for problems found in the System.
User Education	
User Education Architect	Responsible for creating the User Education Strategy and designing the User Education materials.
User Education Developer	Creates the User Education materials.
User Trainer	Delivers System training end users to maximize their benefit and efficiency of the System.

Each role is responsible for the decisions for their portion of the Product. These decisions are captured in their respective artifact. As a result, each section of an artifact is assigned a single owner. Owners are encouraged to collaborate with and delegate to other team members; however, the owner is ultimately accountable for their artifact's timeliness, completeness, and quality.

Artifact ownership scales to massive engineering projects. The Empire State Building had just one lead architect - William F. Lamb. Even though 100s of designers and more than 3,400 workers were employed to build the mammoth structure, he was ultimately responsible for the design. He divided the complexity among other designers, each owning their part, but ultimately integrated the blueprints into a single design.

The alternative to artifact ownership is "design by committee." Agile w/ Blueprints emulates residential and commercial engineering, not the US Congress.

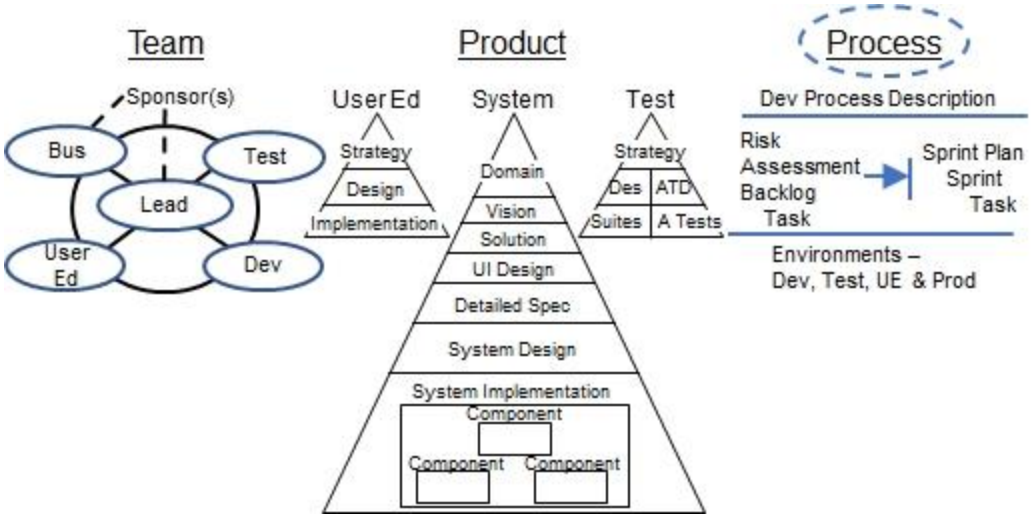
Finally, assigning clear boundaries of responsibility and accountability is more Agile because decisions are made faster. It avoids the "I thought you had that" syndrome. And each owner takes pride in their work because it's their baby.

In our sample eRecipeBox project, Dad fills the project sponsor and business unit roles. Two resources are allocated to build the product – A. Leader and N. Architect. Following are the assigned roles for eRecipeBox MVP 1.0.

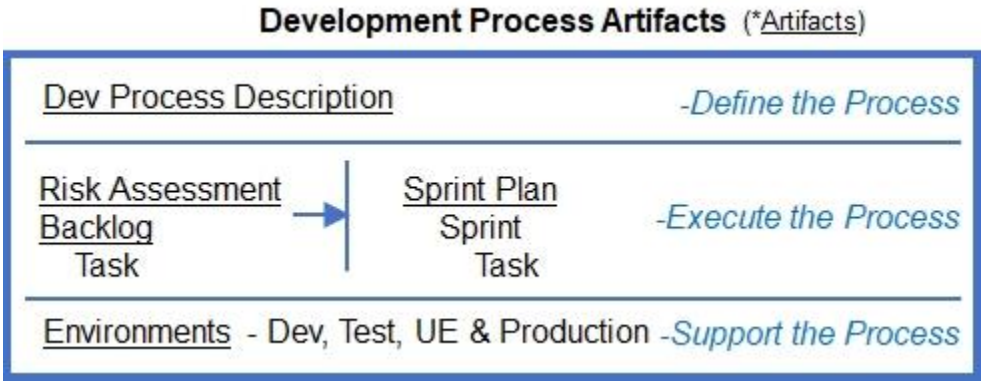
Role	Member	Owned Artifacts
Sponsors		
Business Executive Sponsor	Dad and Mom	
IT Executive	Dad	
Project Champion	Dad	
Business		
Business Unit Manager	Dad	
Lead Business User Representative	Dad	
Business User Representative	Dad	
Subject Matter Expert (SME)	Dad	
Project Leadership		
Team Lead	A. Leader	Risk Assessment, Sprint Plan, Product & Project Backlogs, coach team members that need assistance
Product Owner	A. Leader	Product Vision (which includes Product Roadmap)
System Analyst	A. Leader	Business Model, Solution Model, Detailed Spec (non-UI)
UI Architect	A. Leader	UI Design, Detailed Spec (UI sections)
Development		
System Architect	N. Architect	System Design (except DataStore Design)
Data Architect	N. Architect	DataStore Design
Developer	N. Architect	System Implementation (code), Developer Tests
Graphic Designer	N/A	UI and UE Graphics
System Integrator	N. Architect	System Deployment Package, Automated Build Tools
Test		
Test Architect	A. Leader	Test Strategy, Test Suite Design, Automated Test Design
Test Developer	A. Leader	Test Cases, Automated Tests, Test Data
System Tester	A. Leader	Test Execution Reports, Create issue Tasks
User Education		
User Education Architect	A. Leader	User Education Strategy, User Education Design
User Education Developer	A. Leader	User Education Implementation
User Trainer	A. Leader	

5 Agile w/ Blueprints Process

Team members create Product artifacts, including blueprints and their implementation. This section describes the Process for creating the Product artifacts.

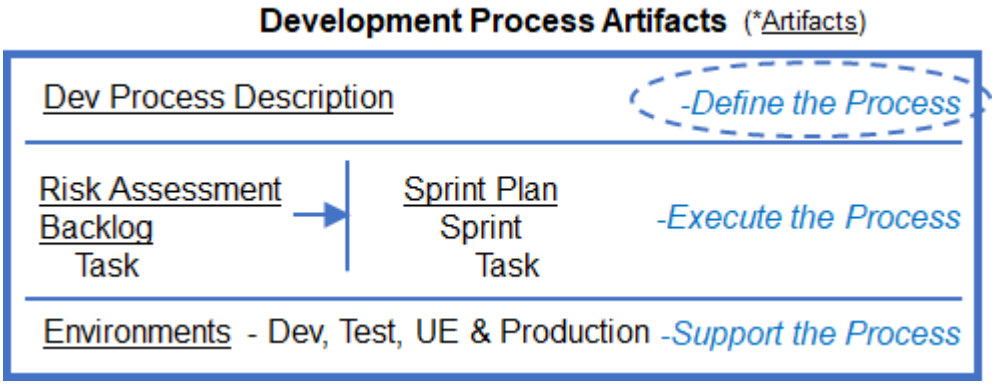


Process artifacts are organized into three views–



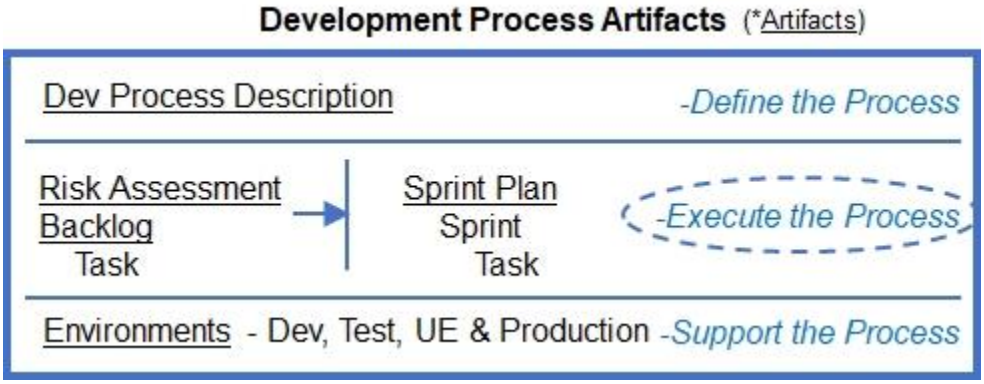
- Dev Process Description – Commonly called Software Development Lifecycle (SDLC)
- Executing the Process – Project management steps, specifically Sprint planning and execution
- Environments that Support the Process – Project Leadership, Development, Test, UE, and Production environments

5.1 Define the Process



eRecipeBox is developed using Agile w/ Blueprints as described in *How I Built a Business App using Agile w/ Blueprints*, available on Amazon for $9.99.

5.2 Execute the Process



To execute AwB's process, the Team Lead assesses Risks, grooms the Backlog, and updates the Sprint Plan with remaining project Tasks. The team completes the Tasks in each Sprint to develop the next Increment.

This section narrates eRecipeBox's Process to develop and deploy MVP 1.0.

5.2.1 Risk Assessment

eRecipeBox MVP 1.0 Risks

Risk	Exposure	
		Technical Risks
SW Architecture/ Commercial Components	10	If DevExpress components don't meet the product needs, we'll need to start over. Specifically, sub-second response w/ 25,000 users in DataStore supporting 500 concurrent users. Will components be supported for 15-20 years? **Mitigation:** -Develop architecture proof of concept early
Appium/ WinAppDriver	5	WinAppDriver is recommended by Microsoft, is open source, and supported by Microsoft. However, it doesn't seem to have a lot of activity in the GitHub repo. **Mitigation:** -Minimize automated test framework code that is unique to Appium/WinAppDriver
Data Security	4	Hackers accessing our internal network which contains sensitive data **Mitigation:** -Apply modern and robust security requirements – user, network and device. -Limit other servers with sensitive data on the production LAN
		Business Risk
Business User Adoption	9	Will the end users actually use eRecipeBox on a daily basis and not revert back to their ad hoc, Man system? **Mitigation:** -Verify UI design with business users early; adding and maintaining recipes is easy and meal planning is easy. Business Rep test drives early Sprint increments and provides usability feedback. -Assist end users to migrate their existing recipes into eRecipeBox -Monitor usage in first weeks of production
		Financial Risk
Total Cost of Ownership (TCO)	8	eRecipeBox dev and maintenance resources exceeds budget. **Mitigation:** -Leverage commercial party components (eg, DevExpress) to minimize development and test. -Develop PoC early (time boxed) so that the team can more accurately estimate TCO. -Evaluate commercial solutions.

5.2.2 eRecipeBox MVP 1.0 - Sprint Plan

Below is the final version of eRecipeBox's Sprint Plan, updated to reflect the actual Project Tasks and events. All eRecipeBox MVP 1.0 artifacts (including implementation code) are available here.

First three Sprints.

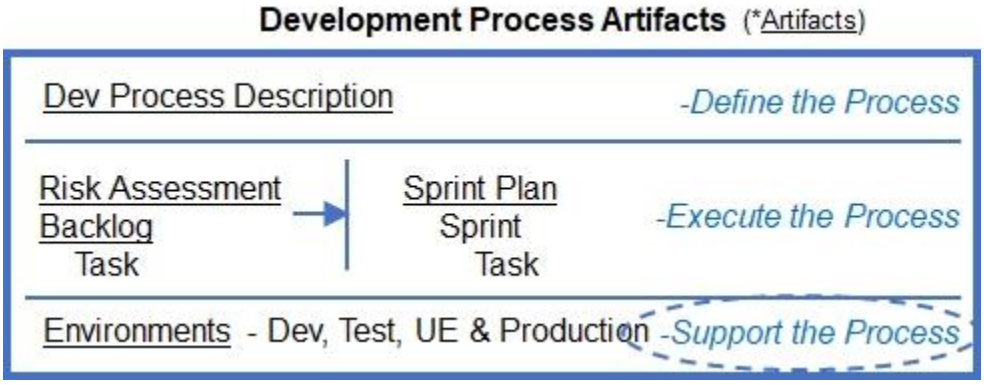
Task Name	Duration	Start	Resource Na
Develop eRecipeBox MVP 1.0	**30 days**	**Mon 1/16/23**	**Lead,Arch**
Sprint 1 - Prod Vision,Soltn Mdl, UI Design, Research system design, Tst Str, Sprint Plan	**5 days**	**Mon 1/16/23**	**Lead,Arch**
Create Domain Model, Product Vision, Solution Model V0.1	1 day	Mon 1/16/23	Lead
Create UI Design V0.1	0.5 days	Tue 1/17/23	Lead
Create sprint plan and estimate project cost V0.1	0.5 days	Tue 1/17/23	Lead,Arch
Analyze legacy recipe data. Research automated testing tools	0.5 days	Wed 1/18/23	Lead
Create Test Strategy, Automated testing tools w/ PoC	2 days	Wed 1/18/23	Lead
Baseline Solution, UI Design & Sprint Plan. Review with Business	0.5 days	Fri 1/20/23	Lead
Evaluate & select 3rd party ORM and UI framework	1 day	Mon 1/16/23	Arch
CRUD PoC with 3rd party ORM and UI framework, 2 tier	2 days	Tue 1/17/23	Arch
Dev backup & restore DB	2 days	Thu 1/19/23	Arch
Sprint 2 - Stabilize System Design, Dev Search/EditRC, Validate Test Strategy & tools	**5 days**	**Mon 1/23/23**	**Lead,Arch**
Evaluate and select recipe website(s) for import	0.5 days	Mon 1/23/23	Lead
Develop and run automated tests against Sprint 1	4 days	Mon 1/23/23	Lead
Update Test Strategy (automation and test data)	0.5 days	Fri 1/27/23	Lead
CRUD PoC, 3-tier using https WCF	0.5 days	Mon 1/23/23	Arch
Add support for SQLite, PostgreSQL	0.5 days	Mon 1/23/23	Arch
Dev business object classes (aka models)	0.5 days	Tue 1/24/23	Arch
Dev Import from AllRecipes.com	1 day	Tue 1/24/23	Arch
Dev SearchRecipeBox, EditRecipeCard V0.1	2 days	Wed 1/25/23	Arch
Log all control names to support automated testing	0.5 days	Fri 1/27/23	Arch
Sprint 3 - Dev Login/Import text,/EditGList/ViewRC. Add bus rules. Detailed Spec. Test Sprint	**5 days**	**Mon 1/30/23**	**Lead,Arch**
Test Sprint 2. Refactor Automated Test Design	3.5 days	Mon 1/30/23	Lead
Review test results with Business. Decide to Add Calendar	0.5 days	Thu 2/2/23	Lead
Add Calendar to System Spec	0.5 days	Fri 2/3/23	Lead
Create Detailed Spec V1.0	0.5 days	Fri 2/3/23	Lead
Dev ImportRecipeCard from Text	1 day	Mon 1/30/23	Arch
Dev business rules, enforce unique IDs	1 day	Tue 1/31/23	Arch
Dev LogIn w/ authentication	1 day	Wed 2/1/23	Arch
Dev EditGroceryList, ViewRecipeCard	2 days	Thu 2/2/23	Arch

Remaining three Sprints.

Task Name	Duration	Start	Resource Na
Sprint 3 - Dev Login/Import text,/EditGList/ViewRC. Add bus rules. Detailed Spec. Test Sprint	**5 days**	**Mon 1/30/23**	**Lead,Arch**
Sprint 3.5 - Dev Calendar/Help/Admin/New lookups, Refactor, Bugs, Test Sprint 3	**5 days**	**Mon 2/6/23**	**Lead,Arch**
Test Sprint 3, including stress test w/ large datastore	2 days	Mon 2/6/23	Lead
Refactor Automated Test Design	1 day	Wed 2/8/23	Lead
Update Test Strategy and Test Suite Design V0.2 - calendar, functional tests	0.5 days	Thu 2/9/23	Lead
Review test results with Business. GPT. Update System Spec.	0.5 days	Thu 2/9/23	Lead
Dev Calendar and GPT tests	0.5 days	Fri 2/10/23	Lead
Dev Calendar	2 days	Mon 2/6/23	Arch
Refactor & polish SimpleCRUDFramework, EditRecipeCard, SearchRecipeBox	1.5 days	Wed 2/8/23	Arch
Dev Help, Admin user	0.5 days	Thu 2/9/23	Arch
Backlog issues found in Sprint 3	**1 day**	**Thu 2/9/23**	**Arch**
Bug - SearchRecipeBox doesn't search ItemDescription	0 days	Thu 2/9/23	
Bug - RecipeCard saved with duplicate Title	0 days	Thu 2/9/23	
...etc	1 day	Fri 2/10/23	
Sprint 4 - Dev GPT, Fix Punchlist, Prep Production, Final Acceptance Test	**5 days**	**Mon 2/13/23**	**Lead,Arch**
Full Sys Test - Bus, Functional, NonFunctional	2 days	Mon 2/13/23	Lead
Test GPT	0.5 days	Wed 2/15/23	Lead
Run final regression and acceptance tests	2 days	Wed 2/15/23	Lead
Develop UE education (video)	0.5 days	Fri 2/17/23	Lead
Dev GPT	1 day	Mon 2/13/23	Arch
Create production env & database	0.5 days	Tue 2/14/23	Arch
Remaining MVP 1.0 backlog and punchlist tasks	3.5 days	Tue 2/14/23	Arch
Bug - Import recipecard parser error 1 (15 oz can) black beans	0 days	Tue 2/14/23	Arch
Bug - App crashes with bad data	0 days	Tue 2/14/23	Arch
Bug - Order of Ingredient is not retained	0 days	Tue 2/14/23	Arch
Enh - Disallow duplicate keyword to RecipeCard	0 days	Tue 2/14/23	Arch
Enh - Enable import buttons only when creating a new recipe card	0 days	Tue 2/14/23	Arch
Bug - Scroll to the previously edited Recipe after closing the form	0 days	Tue 2/14/23	Arch
Enh - Prevent the schedular from changing to DayView mode	0 days	Tue 2/14/23	Arch
etc.	3.5 days	Tue 2/14/23	Arch
Sprint 5 - Deploy MVP 1.0, Update artifacts to As Built	**5 days**	**Mon 2/20/23**	**Lead,Arch**
Deploy MVP1.0 to Production	1 day	Mon 2/20/23	Arch
Update System Design MVP1.0	2 days	Tue 2/21/23	Arch
Update System Spec MVP1.0	1 day	Mon 2/20/23	Lead
Finish test scripts and run FunctionalTestCase01	2 days	Tue 2/21/23	Lead
Develop and run StressTestCase01 against large test data set	1 day	Thu 2/23/23	Lead
Update Test Documentation MVP1.0	1 day	Fri 2/24/23	Lead
eRecipeBox Backlog	0 days	Mon 1/16/23	
Bug - Sporadic error - Flyout window is already disposed	0 days	Mon 1/16/23	
Advanced Search: arbitrary NOT AND OR combinations	0 days	Mon 1/16/23	
Share my eRecipeBox with other users	0 days	Mon 1/16/23	
Print RecipeCard	0 days	Mon 1/16/23	
Develop eRecipeBox mobile app	0 days	Mon 1/16/23	
Bug - Adding Keyword slow w/ large data set	0 days	Mon 1/16/23	
etc.	0 days	Mon 1/16/23	

5.3 Support the Process

Each team needs a work environment to develop their artifacts – Project Leadership, Development, Test, UE, Production (i.e., the Business Team).



5.3.1 Project Leadership Environment

All eRecipeBox documentation is located in

...\eRecipeBoxSampleApp\ eRecipeBoxSystem\ eRecipeBox\ Documentation

Product Leadership owns and maintains these files.

- eRecipeBoxArtifacts.docx, Domain Model, System Spec, Product Vision sections– Master source file for all eRecipeBox Agile w/ Blueprints documentation.
- eRecipeBoxArtifactsFinal.docx – DocumentGenerator generates this final version by inserting requirements documented as annotations in source code into eRecipeBoxArtifacts.docx.
- eRecipeBoxUML.ncp – eRecipeBox UML diagrams.
- eRecipeBoxMVPFinal.mpp – eRecipeBox Sprint plan.
- eRecipeBoxGraphics.pptx – PowerPoint graphics.
- eRecipeBoxGraphics.vsdx - Visio graphics.

Tools used to create and maintain eRecipeBox documentation.

- Microsoft Office 2019 or greater – Word, Excel, Visio and PowerPoint
- Microsoft Project – Sprint Plan
- NClass 2.8.2 – Freeware UML diagraming tool.

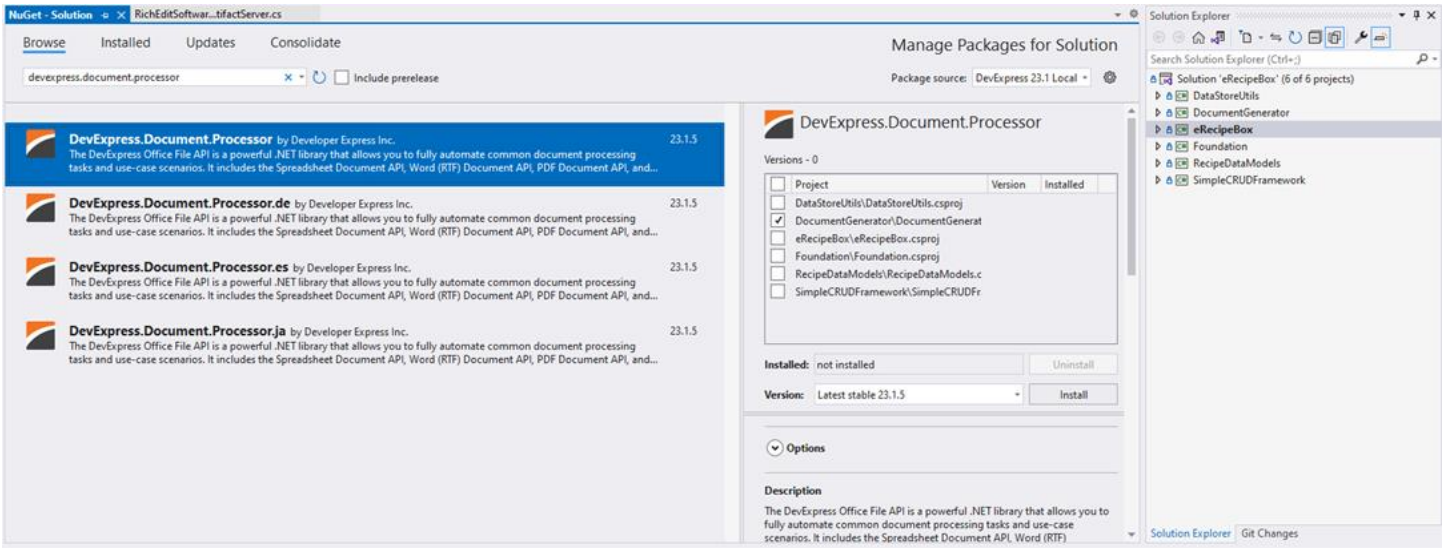
The DocumentGenerator project generates the final version of the artifacts document by inserting detailed business rule and UI form requirements into eRecipeBoxArtifacts.docx.

DocumentGenerator

- Microsoft.CodeAnalysis
- DevExpress.Document.Processor V2x.x.x

DevExpress.Document.Processor is licensed, so set your Package source to their licensing URL.

Use NuGet Packages to Install Office File API Components | Office File API | DevExpress Documentation



5.3.2 Dev Team Environment

This section describes the tools and installation steps to build and debug eRecipeBox implementation. It has only been tested with a new, clean Windows PC.

System Requirements: Windows 10 or 11 with at least 16GB RAM. [Example: KAMRUI AK2 Plus Mini PC 3.4GHz Micro Computer, 16GB RAM 500GB SSD at ~$170 for Dev and System test.]

Highly recommend two monitors: Run the eRecipeBox app on one and Visual Studio on the other.

Steps to build a dev environment. Follow along with this video. Pause at each step so stay in sync.

Install Visual Studio 2022

1. Visual Studio 2022 Community Edition – Download Latest Free Version (microsoft.com)
2. Login to visual studio with your Microsoft account
3. Select .net desktop development and Data storage and processing.
4. Verify installation by creating a .NET framework WinForms app and running it.
5. Close Visual Studio.
6. Enable Code Cleanup on Save.

Install DevExpress DXperience 30-day trial

1. DXperience Subscription: .NET and JavaScript Development | DevExpress
2. Select WinForms, XAF, Reporting, Office File API.
3. Verify it is installed correctly by creating a DevX Ribbon form app, add a grid and run it.
4. Close the app.
5. Close Visual Studio.

Note: If you need more than 30 days to get through the Quick Start, you may submit a request to DevExpress support to extend your free trial.

Download and unzip eRecipeBox source package

1. Download and unzip eRecipeBox package into your training project directory. The version must match the DevExpress DXperience version.

Install ToggleRegions extension

eRecipeBox uses #regions extensively. Install this VS extension which collapses all #regions with Ctl-r Ctl- - (control minus)

1. Download CollapseRegionVS2022.vsix.
2. Close all Visual Studio solutions.

3. Run the VSIX file.

Verify it works. Open eRecipeBox.sln. Ctl-m Ctl-o collapses all methods. Then Ctl-r Ctl- - (control minus) collapses all regions. Note: In testing, I experienced different behavior with different installations. Play with Ctl-m Ctl-p to expand all, Ctl-m Ctl-o to collapse all (sometimes methods, sometimes regions) or use Ctl-r Ctl- - (control minus) to collapse all.

Install SQL Server Developer Edition 2022 & SSMS 19

1. Download SQL Server 2022 installer, Developer edition.
2. Select Basic
3. Select Install SSMS 19
4. Download SSMS
5. Close SQL Server Developer Edition installer
6. Reboot PC
7. Install SSMS
8. Configure SQL Server 2022 – configuration manager
9. Verify SQL service is running
10. Enable named pipe and TCP/IP network client protocols
11. Restart SQL Server Service (MSSQLSERVER)
12. Close configuration manager
13. Run SSMS
14. Connect to SQL Server using windows authentication
15. Verify installation by creating a database and a table.
16. Open server engine properties
17. Set server authentication = SQL Server and Windows Authentication mode
18. Restart MSSQL service for the changes to take effect.

Install PostgreSQL 16.x, Optional

eRecipeBox can use SQL Server, PostgreSQL or SQLite as its DataStore provider. If you plan to use PostgreSQL, follow these steps.

1. Download PostgreSQL.
2. Run the installer, selecting default options and providing a password for superuser (postgres)
3. The installer also installs the pgAdmin app. Launch it and connect using the postgres superuser password.
4. Verify successful installation by viewing the default database.
5. We need a populated eRecipeBox PostgreSQL database. Use eRecipeBox to restore a PostgreSQL backup to your newly installed PostgreSQL server….
6. Launch eRecipeBox (currenting pointing to SQL Server)
7. On Admin tab, click Restore Database
8. Copy the PostgreSQL's "Recipes" connection string value from eRecipeBox app.config (starting with `XpoProvider=Postgres`) into the "New Schema XPO Connection String" textbox. Replace the password placeholder with your password (NOT ENCRYPTED).
9. To select the Restore Folder parameter, click the ellipse button and select the xxxx_SysTest01_PostgreSQL folder.
10. This creates a new PostgreSQL database and loads the system test data.
11. Before closing eRecipeBox, use the Admin tab to encrypt your postgres superuser password. You'll need the encrypted password for the app.config file.
12. To verify, return to pgAdmin and refresh Databases. View list of tables in "recipeTest1.public" schema.
13. Return to eRecipeBox app.config.
14. Rename SQL Server connection name to "RecipesSS" and PostgreSQL connection name to "Recipes"
15. Run eRecipeBox.

16. Return to previous state. Rename SQL Server connection name to “Recipes” and PostgreSQL connection name to “RecipesPG”.

Install DB Browser for SQLite

1. Download DB Browser for SQLite
2. Run the installer and select default options

Open eRecipeBox.sln, build and run

1. Open …\eRecipeBoxSampleApp\ eRecipeBox.sln
2. Build Solution
3. If you get a clean build, click Start to run eRecipeBox in the debugger
4. Should be up and running.
5. Review app.config file, <appSettings>.
6. At this point, you should be able to run the app in the debugger and step through the code as you review the System Design.
7. Verify email using Google Service by emailing a Grocery List
8. Once you’ve verified you’ve successfully built a dev environment, run a backup (Admin.Backup Database).
9. Create a local GitHub repository. (or cloud repo if working with a team)

5.3.2.1 Configure 3rd Party Services

Create and configure Google Cloud Gmail API service

eRecipeBox uses Google Cloud services to send GroceryList emails.

Detailed Steps:

1. Create Google Cloud project. This Google account pays for the service (free with low usage).
2. Enable Gmail API service with OAuth2 authentication
3. Configure OAuth consent screen for desktop app. Grant authorization to Google users to use the service in order to send emails. Google authenticates users when emailing a GroceryList. Note that once authenticated, eRecipeBox can email a GroceryList to any email.
4. Create API key for the client to use the Gmal API service (embedded in gCredentials.json file).
5. Add the credentials to eRecipeBox project. Replace contents of …\eRecipeBoxSampleApp\ eRecipeBoxSystem\ Foundation\ gCredentials.json with your credentials.
6. Test: Run eRecipeBox and email the GroceryList.

Create and Configure OpenAI API service

eRecipeBox uses OpenAI service to import RecipeCard from GPT.

Detailed Steps:

1. Create OpenAI API account.
2. Create secret API key
3. Use eRecipeBox Admin to encrypt the key and set the encrypted key in app.config.
4. Run eRecipeBox and import from GPT to test.

Create and Configure Azure Speech to Text Service

eRecipeBox uses Azure SpOpenAI service to import RecipeCard from GPT.

Detailed Steps:

1. Create Azure account
2. Create Resource Group

3. Create Speech Service
4. Obtain API Key
5. Use eRecipeBox Admin to encrypt the key and set the encrypted key & region in app.config.
5. To test, run eRecipeBox and import from GPT, using speech.

5.3.2.2 Multi-tier Dev Environments

5.3.2.2.1 Install SQL Server on another LAN PC

These additional steps are required when installing SQL Server on another LAN PC. We need to ensure client PC can communicate to SqlServerPC and establish a connection with SQL Server.

1. Set SqlServerPC settings.network & internet.<network>.properties.network profile type = Private
2. Turn off firewall for the Private network profile (check firewall status)
3. Ping <SqlServerPC>
4. Set "RecipesLanSQLServer" connection string parms. Encrypt the password. Rename it to "Recipes".
5. Run eRecipeBox client. Close eRecipeBox.
6. On SqlServerPC, turn On firewall for the Private network profile and create rule allowing inbound connections on port 1433.
7. Run eRecipeBox client.

5.3.2.2.2 Install RecipeBoxDataService on Dev PC

To deploy eRecipeBox in a 3-tier configuration, we need to install the RecipeBoxDataService and configure eRecipeBox client to use the service rather than connecting direction to the DataStore.

1. Create a self-signed certificate by updating and running CreateCert.ps1 script. This script does the following.
    a. Create a self-signed certificate.
    b. Bind the certificate to IP address and port 5757
    c. Add certificate to TrustedPeople
2. Run RecipeBoxDataServiceMain
3. Configure eRecipeBox client to use the service. In app.config, set UseWCF = true.
4. Run eRecipeBox

5.3.2.2.3 Deploy RecipeBoxDataService on another PC

These additional steps are required when deploying RecipeBoxDataService on another LAN PC. We need to ensure client PC can communicate with the service.

1. Set AppPC settings.network & internet.<network>.properties.network profile type = Private
2. Turn off firewall for the Private network profile (check firewall status)
3. Ping AppPC
4. Deploy RecipeBoxDataServiceMain binaries to AppPC
5. Create a self-signed certificate as described in Section 8.3.2.2.2.
6. Ensure AppPC can successfully connect to SQL Server
7. Set RecipeBoxDataServiceMain app config file, `Recipes` connection string to recipeTest1 database.
8. Run RecipeBoxDataServiceMain.exe to start the service
9. Open web browser and browse https://<AppPC>:5757/service
10. Set eRecipeBox app config setting UseWCF = true.
11. Set eRecipeBox app config connection RecipeService = https://localhost:5757/service
12. Run eRecipeBox client. Close eRecipeBox.
13. Turn On firewall for the Private network profile and create rule allowing inbound connections on port 5757
14. Then run eRecipeBox client.

5.3.3 Test Team Environment

System Requirements: Windows 10 or 11 with at least 16GB RAM.

Install and configure WinAppDriver, restore system test database, then run automated tests. These steps start at 27:40 in the dev & test environment installation video.

Install WinAppDriver

1. Instructions for installing WinAppDriver
2. Download WinAppDriver and install it.
3. Verify WinAppDriver was installed in C:\Program Files\Windows Application Driver
4. Enable Developer Mode for your Windows PC.

Verify test environment by running BusinessTests.Business01 automated test

1. Prerequisites: SysTest01 database loaded in SQL Server. Note that WinAppDriver automated testing doesn't work using SQLite (file lock issue), so we need to create a schema in SQL Server.
2. Restore a recipeTest1.bak file into SQL Server.
3. Return to the eRecipeBox solution, modify the app.config to refer to the new restore by renaming the SQLite connection string to RecipesSQLite and renaming the SQL Server connection string to Recipes. Run the app in the debugger to verify it works.
4. Close the app and return to the eRecipeBoxSystemTests solution.
5. Open the Test Explorer
6. Select BusinessTests.Business01 and debug it. You should see this.
7. Note: The system test needs to start with a known DataStore, so always restore recipeTest1.bak before running BusinessTests. BusinessTests. TestCase01 and FuntionalTests. SearchRecipeBox. TestCase01.
8. Create a local GitHub repository. (or cloud repo if working with a team)

5.3.4 UE Team Environment

Tools used to create the User Education "how to" video when the user clicks "Help"

- Screen recording - Debut Video Capture and Screen Recorder Software
- Video editing - Filmora

5.3.5 Production Environment

5.3.5.1 Install eRecipeBox

Install eRecipeBox Client:

Run eRecipeBoxSetupV2x.x.exe. The eRecipeBox client app is installed in this folder -

C:\Users\<user name>\ AppData\ Local\ Programs\ eRecipeBox.

Install eRecipeBox DataStore Service on app server:

Login to a Windows Administrator account.

Run eRecipeBoxDSServiceSetupV2x.x.exe

eRecipeBoxMain is installed in this folder

C:\Users\<user name>\ AppData\ Local\ Programs\e RecipeBoxMain

Install SQL Server or PostgreSQL on database server:

Follow the same steps in Dev Team Environment to install the database server (SQL Server or PostgreSQL).

5.3.5.2 Configure eRecipeBox

eRecipeBox can be deployed in 1, 2, or 3-tier configurations.

5.3.5.2.1 Local, single-user configuration

Local, single user configuration stores all RecipeCards in a local, SQLite database file.

Replace the contents of C:\Users\<user name>\AppData\Local\Programs\eRecipeBox\eRecipeBox.exe.config with AppSingleUser.config.

Use the following Recipes connection setting

```
<add name=”Recipes” connectionString=”XpoProvider=SQLite;Data
Source=SQLiteData\recipeTest1.sqllitedb” />
```

5.3.5.2.2 Two-tier configuration

2-tier config connects to a database on the LAN.

Edit C:\Users\<user name>\AppData\ Local\ Programs\ eRecipeBox\eRecipeBox.exe.config

Ensure UseWCF is false

Use the following Recipes connection setting if DB is installed on same PC as the client app.

```
<add name="Recipes" connectionString="XpoProvider=MSSqlServer;Server=localhost;initial
catalog=recipeTest1;Trusted_Connection=True" />
```

Use the following Recipes connection setting format if DB is installed on LAN. Update IP address, user id, and password.

```
<add name="Recipes" connectionString="XpoProvider=MSSqlServer;Data Source=192.168.1.28;initial
catalog=recipeTest1;user id=<userid>;password=7B01B51AB4F005BFF735FFAE00A;Persist Security
Info=true" />
```

To encrypt the PW, run the app in local, single user mode. Then click Admin->Encrypt App Setting.

5.3.5.2.3 Three-tier configuration

eRecipeBox.exe client config. Access DataStore through RecipesService.

Client configuration.

Edit C:\Users\<user name>\AppData\ Local\ Programs\ eRecipeBox\ eRecipeBox.exe.config.

Ensure UseWCF is true.

Use the following RecipesService connection setting if running on same PC

```
<add name="RecipesService" connectionString="https://localhost:5757/service" />
```

Use the following RecipesService connection setting if running on LAN. Update IP address.

```
<add name="RecipesService" connectionString="https://192.168.x.x:5757/service" />
```

Use the following RecipesService connection setting if running on WAN. Update your URL.

```
<add name="RecipesService" connectionString="https://www.yourDomain.com:5757/service" />
```

App Server configuration.

In RecipeBoxDataServiceMain app.config file -

Use the following Recipes connection setting if DB is installed on same PC as the client app.

```
<add name="Recipes" connectionString="XpoProvider=MSSqlServer;Server=localhost;initial
catalog=recipeTest1;Trusted_Connection=True" />
```

Use the following Recipes connection setting format if DB is installed on LAN. Update IP address, user id, and password.

```
<add name="Recipes" connectionString="XpoProvider=MSSqlServer;Data Source=192.168.1.28;initial catalog=recipeTest1;user id=<userid>;password=7B01B51AB4F005BFF735FFAE00A;Persist Security Info=true" />
```

To encrypt the PW, run the app in local, single user mode. Then click Admin->Encrypt App Setting.

In order for internet (non-LAN) clients to open connections, you will need to open port 5757 and route traffic through your firewall.

==========

IMPORTANT WARNING!!

As deployed,

1. Client WCF connections currently accept self-signed certificates. Ensure to obtain a proper certificate and remove the System.Net.ServicePointManager.ServerCertificateValidationCallback in DataStoreUtils. XpoService.GetWSHttpDataStore to enforce properly signed certificates.
2. Implementation only checks for a valid email. Update RecipeBoxDataService. CustomUserNameValidator. Validate and eRecipeBox. DataStoreServiceReference. DataStoreServiceReference. UpdateUserProfileUponLogin logic to also validate Windows client system name and perhaps MAC address and client Windows user name to be more restrictive.

==========

Data Server configuration.

Before releasing to production, create a database server account for eRecipeBox app server to connect. Ensure it has public and sysadmin server roles (SQL Server) and full access to the Recipes database.

Follow these steps to enable incoming SQL Server port 1433 connection requests.

6 Hands on Exercises

Engineers develop their skills by doing. An engineer's first Agile w/ Blueprints experience should be in a play sandbox, not a mission-critical business app. Also, it's easier to fix and enhance a working app rather than start from scratch. eRecipeBox Source Package and GitHub link can be found here.

This section provides ~45 eRecipeBox product backlog tasks. They're grouped by type and functional area. There are minimal dependencies among them, so you can implement the ones that interest you most. Start with a few small backlog tasks to build your confidence, then work your way to large tasks.

Teams can develop their skills and proficiency by working on these together. Assign team members to their roles. Each role updates their assigned AwB blueprints.

I did not repeat the eRecipeBox Product tasks in the Product Roadmap, so they also are a source of hands-on exercises.

Additional backlog tasks can be found by searching the source code for these annotations.
#TODO
#TODO BUG
#TODO PERFORMANCE
#TODO ENHANCEMENT
#TODO REFACTOR

As you work through your Sprints, update eRecipeBox blueprints as you make your decisions. The source blueprints for MVP 1.0 are located in \ eRecipeBoxSampleApp\ eRecipeBoxSystem\ eRecipeBox\ Documentation.

If you specify and implement new, creative eRecipeBox enhancement(s), I'd like to hear about them. Email me at feedback@agilewithblueprints.com with your creations.

6.1 Functional Enhancements

Functional Area	Size	Impacts	Backlog Task
Cooked Dish	Lrg	Usability	Provide suggestions when meal planning. E.g., "Rule: I want to rotate my 10 rated main dishes every 3 months and eat at least three different cuisines a week."
Grocery List	Small	Usability	Select any number of ingredients while in EditRC or ViewRC, add them to GroceryList.
Grocery List	Small	Quality	When Gmail authorization fails (e.g., user not authorized) present helpful error message and return to GroceryList.
Keyword	Lrg	Usability	Create categories for Keywords to support things like Cuisine (Mexican, Mediterranean, Italian, etc) and Department (Dairy, Cooking, Canned goods, etc.)
Keyword Cooked Dish	Small	Maint.	Fix concurrency vulnerabilities; two sessions can add the same Keyword to an RC. And two sessions can add two CookedDishes on same day.
Logging	Med	Maint.	Enhance app runtime logging; capture errors and send severe errors (and their logs) to a server assist in triaging problems.
Recipe Card	Lrg	Import	Add nutrition data to RecipeCards.
Recipe Card	Lrg	Import	Support RecipeCard images
Recipe Card	Lrg	Bulk Import	Provide power import and export; sets of RecipeCards at once
Recipe Card	Med	Usability	Support "scale ingredients 1x 2x 3x"
Recipe Card	Med	Usability	Support referencing RecipeCards within RecipeCards. eg, Salad recipe references a dressing recipe. Click on the dressing recipe to open it. Also, automatically add CookedDish for the dressing recipe when adding the Salad CookedDish.
Recipe Card	Med	Usability	Allow family members to rate RecipeCards
Recipe Card	Med	Import	Enhance Ingredent Parser. Better support (10.75 ounce) or (10 oz pkg). Research model solutions like https:// github.com/ NYTimes/ ingredient phrase tagger. Search "#TODO ENHANCEMENT Might be able to improve our parse alg" in the code.
Recipe Card	Med	Usability	Enhance Ingredient.SuggestItem. Seach "#TODO ENHANCEMENT SuggestItem" in the code.
Recipe Card	Med	Import	Automatically detect and support all websites that support https:// schema.org/ Recipe. Search "#TODO ENHANCEMENT schema.org" in the code.
Recipe Card	Med	Import	Add more sophistication when importing from GPT. Select 1..n RCs, create a prompt that includes the RC titles and RC ingredients, and request a dish that pairs with it.
Recipe Card	Med	Import	Add more sophistication when importing from GPT. Request 'n' recipes to be returned and allow the user to browse and select which one(s) to import.
Recipe Card	Small	Import	Improve ImportTextRecipeCard to import Instructions: similar to description (eg, all lines up til <end instructions>).
Recipe Card	Small	Import	Upgrade to use the latest version of OpenAI service.
Recipe Card	Small	Import	When importing TextRecipeCardImporter, ignore nondisplayable characters (e.g., bullets). Example: https:// www.liveeatlearn.com/ 1 pan vegan ratatouille/ #wprm recipe container 9608
Recipe Card	Small	Reports	Generate report on RecipeCards and the nutrition in my diet.
Recipe Card	Small	Import	When importing GPT, add a progress bar and a cancel button.

Recipe Card	Small	Import	TextRecipeCardImporter, support importing Keywords. Search "#TODO ENHANCEMENT Import keywords" in code.
Search Recipe Box	Small	Performance	Bind a server mode view to the Schedular to provide efficient scrolling and best performance. Search "#TODO ENHANCEMENT Bind DataStorage.Appointments to a server mode scrolling view" in the code.
Search Recipe Box		Usability	Support Advanced SearchRecipeBox Support arbitrary And, Not, Or
User Profile	Small	CRUD	Allow user to maintain their UserProfile.

6.2 Deployment Backlog Tasks

Size	Impacts	Backlog Task
Med	Maint.	Add autoupdate app version to eRecipeBox. Launch eRB, authenticate, check to see if there is a new version and prompt user to update.
Small	Usability	Install RecipeBoxDataService as a Windows service on a dedicated app server.
Small	Security	Obtain an SSL/TLS certificate for RecipeBoxDataService app server.
Small	Security	Create separate database IDs, dedicated to eRecipeBox app, with proper permissions. Don't use postgres and sa user accounts.
Small	Security	eRecipeBox client app currently installs in C:\ Users\ <user name>\ AppData\ Local\ Programs\ eRecipeBox. Move app data to different location and allow for all users on the PC to run the app. Properly set permissions to app data.
Small	Maint.	Upon app startup, check ModelVersion compatibility in the app against the server and error out if they are incompatible.
Small	Maint.	Package eRecipeBox app into a Docker container

6.3 Design Backlog Tasks

Size	Impacts	Backlog Task
Medium	Import	Move Gmail and GPT interfaces to the app server. Potentially Azure speech to text service as well.
Medium	Maint.	Deploy app server to a cloud service provider (GCP, AWS Azure).
Small	Import	Refactor RecipeCardImporters. Remove all UI dependencies in the importers and move them all to Recipe Box Solution Model. Search for "#TODO REFACTOR Remove all UI dependencies in the importers" in the code.

6.4 Security Backlog Tasks

Size	Impacts	Backlog Task
Med	Auth'n	Add Google, Facebook, Microsoft OAuth2 Authentication options
Med	Auth'n	Force user to verify their email by sending an email to the user and having them click a web link to verify it, before allow them to email GroceryList.
Med	API Keys	Add an Authentication service that provides GPT and Azure speech service API keys, removing the keys from client app.config.
Small	Auth'n	eRecipeBox currently uses two-way encryption for sending Authentication credentials. Store one-way hash (sha256) in the DataStore instead to make it more secure.

6.5 Test Backlog Tasks

Size	Impacts	Backlog Task
Med	Quality	Complete all System Test Suites. Write and execute Test Cases for Search Recipe Box 02, Edit Recipe Card 01, Security 01, and Non Functional 01.
Med	Quality	Design, implement, and execute Test Cases for the remaining eRecipeBox forms.

[i] Support Ticket T687329 “We did not write special code to support testing our WinForms controls by using Appium/WinAppDriver. Since our controls have their own structure, it is unlikely that tests written by mentioned tools will operate properly for them.”