

Programming III – Fall 2024

Course Project: Restaurant Menu

Team Information

Team name, team members' names and IDs.

Antonina Kosyakova - 2238996

Alaa Kirdi - 2363678

Project Description

Describe the application you are building and its features.

Application Overview

This application enables restaurant customers to place orders and allows the manager to manage the menu, view sales. It integrates a user-friendly interface for both customers and restaurant managers.

Key Features

1. Customer Order Placement

- **Food Menu Display:** The menu is categorized into sections like "Starters," "Main Dishes," and "Desserts," with item names, descriptions, prices, and images.
- **Add to Order:** Customers can select items and add them to their order using quantity selectors.
- **Order Summary:** A cart-style section shows selected items and total price.
- **Place Order Confirmation:** Once an order is placed, a "Thank You" screen appears confirming the order.

2. Manager Login

- **Login Portal:** The manager can log in using a username and password to access administrative features.

3. Menu Management

- **Add Items:** Managers can add new items to the menu by specifying name, description, price, and an image.
- **Edit Items:** Existing menu items can be edited or updated (name, description, price, or image).
- **Food Menu Overview:** Managers can view the full menu with options to edit or update items.

4. Sales Report

- **Daily Sales:** The application calculates and displays the total sales for the day.

Purpose of the Application

This system streamlines restaurant operations by:

- Providing customers with an easy and engaging way to order food.
- Allowing managers to efficiently manage the menu and monitor daily sales.

Development Approach

Explain how did you prepare for the project. You can use the 5 steps of algorithmic thinking to you help build this section (you will need to elaborate on each step).

1. Understanding the Problem

In this step, the focus was on comprehending the requirements and challenges faced by both customers and restaurant managers:

- **Customer Needs:**
 - A user-friendly interface for browsing menu items.
 - Ability to place orders with clear visibility of selected items and total cost.
 - Immediate feedback upon successful order placement.
- **Manager Needs:**
 - A secure system for accessing administrative features like managing the menu.
 - Tools for adding, editing, and removing menu items.
 - A dashboard to monitor daily sales.

The primary goal was to create a centralized system that bridges the needs of customers and restaurant managers, making the ordering and management processes seamless and efficient.

2. Formulating the Problem

The problem was broken into smaller, manageable sub-problems to ensure clarity:

1. **Customer Interface:**
 - How to design a menu interface that categorizes food items (e.g., starters, main dishes, desserts).
 - How to enable a cart-like order system that tracks selected items and calculates the total price.
2. **Manager Functionality:**
 - How to implement secure login for managers.
 - How to create options for adding, editing, and viewing menu items dynamically.
 - How to display the total daily sales in a visually intuitive way.
3. **System Features:**
 - How to structure the application to handle real-time updates to menu and order data.

- How to ensure smooth communication between customer and manager functionalities.
-

3. Developing the Application/Algorithm

Here, the problem was transformed into logical workflows:

1. Application Workflow:

- Start with a **home screen** offering two roles: Manager or Customer.
- For **customers**, build a navigation system for the menu and order placement, culminating in a confirmation screen.
- For **managers**, create a login system leading to tools for menu management, order tracking, and sales reporting.

2. Algorithms:

- **Order Calculation Algorithm:**
 - Compute the total order price by summing the price of all selected items multiplied by their quantities.
 - **Sales Report Algorithm:**
 - Aggregate the prices of all orders placed during the day.
 - **Menu Management Algorithm:**
 - Use CRUD (Create, Read, Update, Delete) operations for adding, editing, and deleting items.
 - **Authentication Algorithm:**
 - Validate manager credentials before granting access to admin functionalities.
-

4. Implementing the Application/Algorithm

The implementation was done using **Visual Studio 2022**, with C# as the back-end programming language and **XAML** for designing the UI. Below are the steps:

Setting up the Environment

1. Development Environment:

- Installed and set up Visual Studio 2022 with the required tools for C# and XML development.
- Selected **WPF (Windows Presentation Foundation)** for building a desktop-based application.

2. Data Models:

- Defined classes in C# to represent the core data structures:
 - **MenuItem**: Represents food items with properties like Name, Description, Price, and ImagePath.
 - **Order**: Represents customer orders with properties like OrderID, Items, and TotalPrice.
 - **Manager**: Represents manager credentials with Username and Password.

5. Testing

Thorough testing ensured the application functions as intended:

1. **Unit Testing:**
 - Test individual components like menu display, cart functionality, and manager login.
2. **User Testing:**
 - Simulate customer orders and manager activities to ensure a smooth user experience.
3. **Bug Fixing:**
 - Resolve any issues discovered during testing, like incorrect calculations, interface glitches, or login failures.

OOP Design

Talk about the classes you need to create for the application and what is the purpose of each class. Include the UML class diagram in this section. The UML class diagram should include the relations between the created classes. Do not mention the WPF classes (Window, etc.)

Class	Purpose
Manager	Represents a manager with login credentials.
MenuData	Stores lists of menu items categorized into starters, main dishes, and desserts for saving/loading.
MenuItem	Represents a menu item with details like name, description, price, and image.
MenuManager	Manages the menu items, handles CRUD operations, saves/loads menu data, and tracks daily sales.
Order	Represents a customer order with a list of items and their total price.
OrderItem	Represents an item in an order, including quantity and the total cost for that item.
RelayCommand	Provides a reusable command implementation for handling UI interactions (e.g., button clicks).

Contributions

What did each team member do?

We both collaborated on all aspects of the project equally. Every step, from planning to implementation and testing, was carried out as a team to ensure that the workload was balanced and ideas were shared. Specific contributions include:

- **Planning and Designing:**
 - Brainstormed features and created the project's architecture together.
 - Collaboratively designed the UI layout and workflow.
 - **Development:**
 - Both worked on writing and refining the classes, dividing code writing and review tasks.
 - Alternated between front-end (XAML) and back-end (C#) development to build a seamless application.
 - **Testing:**
 - Shared responsibilities for testing individual features and the overall application functionality.
 - Discussed bugs and implemented solutions as a team.
-

How was the work in the project divided?

The work was divided in a collaborative manner, with both of us contributing equally across tasks:

1. **Back-End Development:**
 - Both team members worked together on designing and implementing the logic for:
 - Managing menu items (MenuManager, MenuItem classes).
 - Processing orders (Order, OrderItem classes).
 - Authentication logic (Manager class).
2. **Front-End Development:**
 - Designed the UI collaboratively using XAML.
 - Worked on binding the front-end with the back-end code in C#.
3. **Testing and Debugging:**
 - Both team members tested features independently to identify issues.
 - Debugged errors together, leveraging two perspectives to find solutions.

App Setup

Third-Party Tools

The following third-party tool was used during the development of the application:

- **ChatGPT:**
 - Assisted in creating detailed documentation for the project.
 - Provided sample data for menu items, such as names, descriptions, and prices, to enrich the application.
 - Helped refine the formal tone and structure of the project report.

Built-In Accounts

The application includes one built-in manager account for authentication:

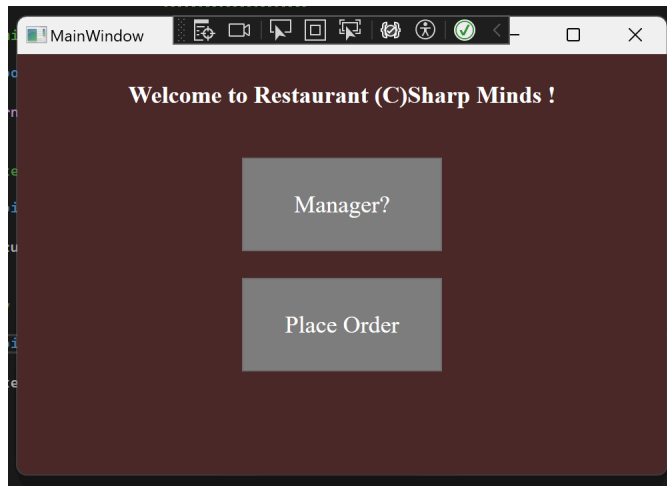
- **Username:** admin
- **Password:** password123

This account grants access to manager-specific features such as menu management, viewing sales, and tracking orders.

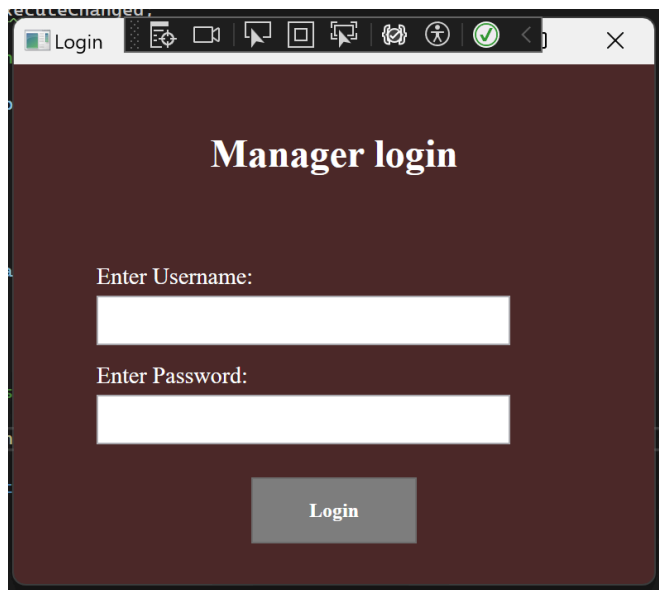
App Snapshots

This section includes snapshots of the final application showing different features. It could be a guideline to using the application. You may include snapshots of the app while being developed. Remember to add explanatory captions to the snapshots.

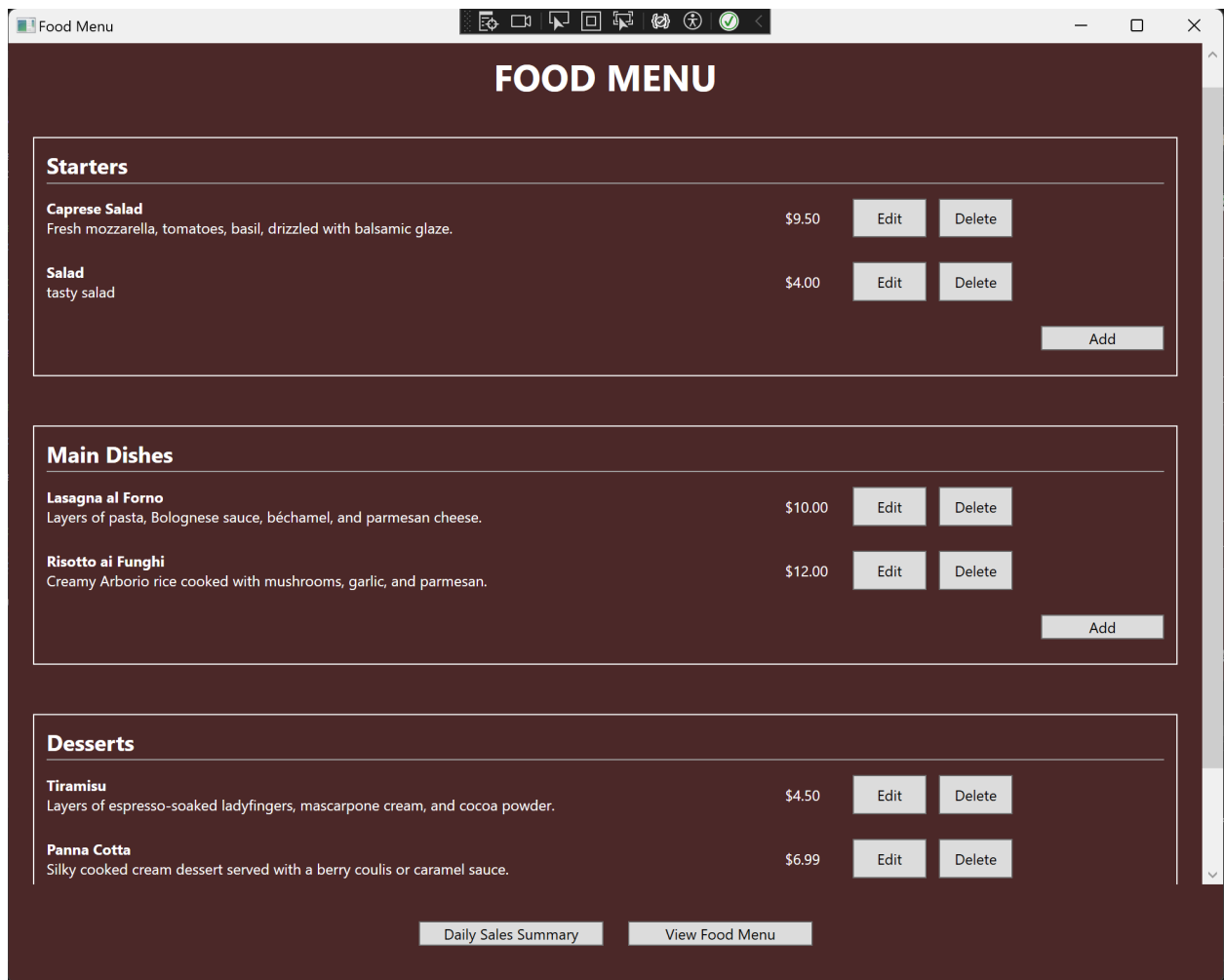
This is the entry point of the application. It asks the user to choose what they want to do: place an order or access manager window.



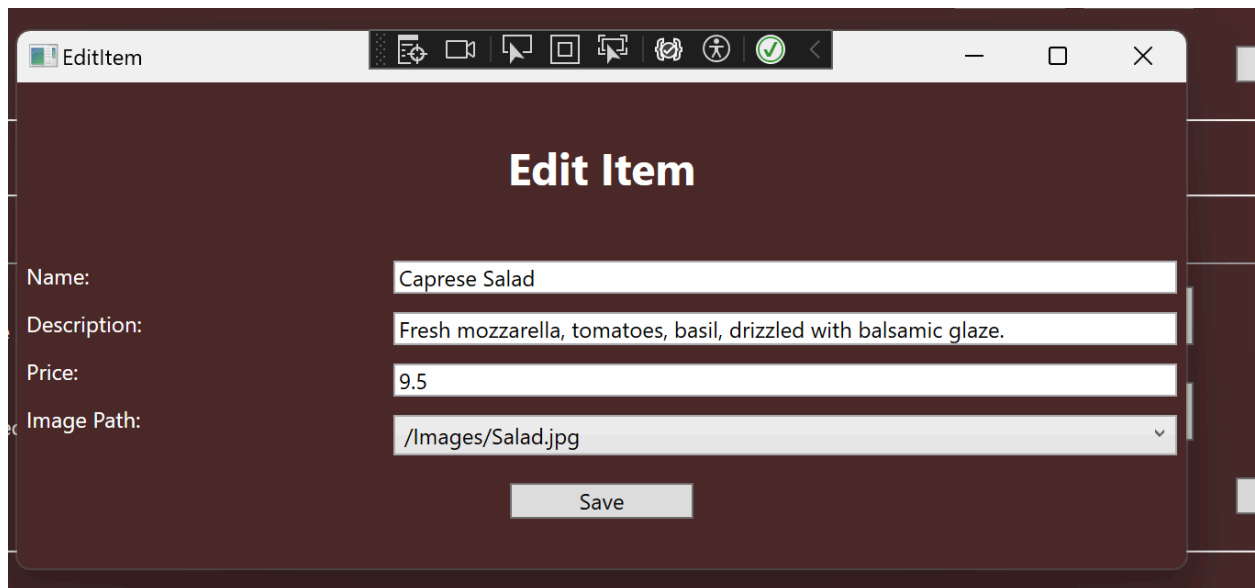
This is the Manager's login, you have a username and a password to enter, if you enter incorrectly it would give you an error message.



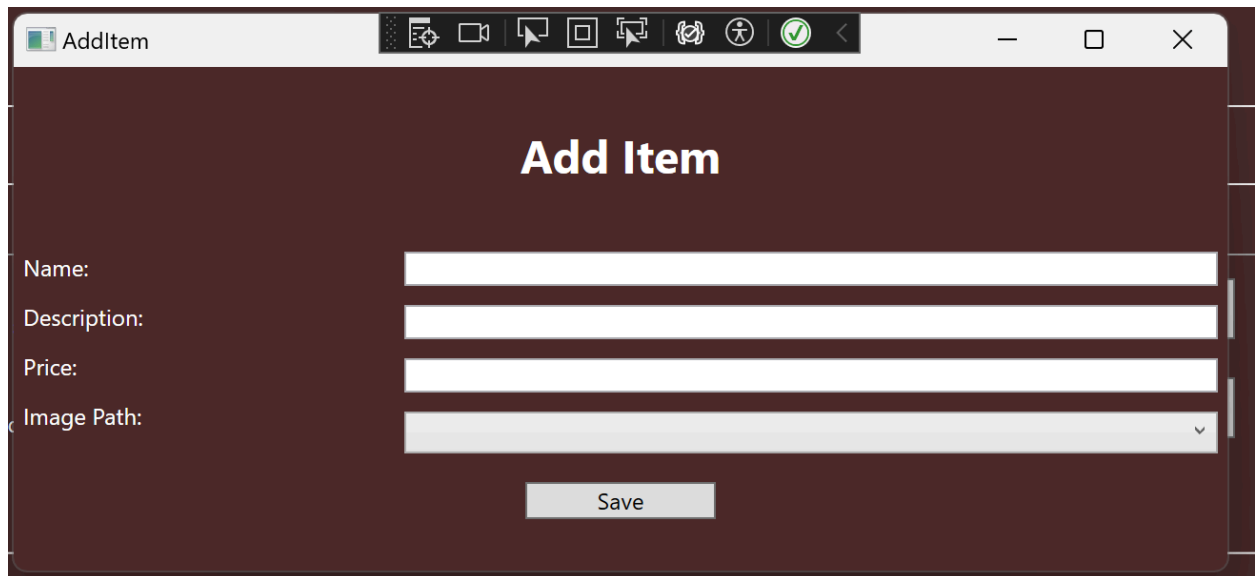
This is the manager window where they can see the current food menu their restaurant has and edit/delete/add any menu items as necessary and view the daily sales summary as well as go directly to the customer side of the food menu to see how it will look for them after the manager makes the changes.



This is the EditItem Window when you press on an Edit Button from above screenshot, It allows you to edit the content of an existing dish and save it.

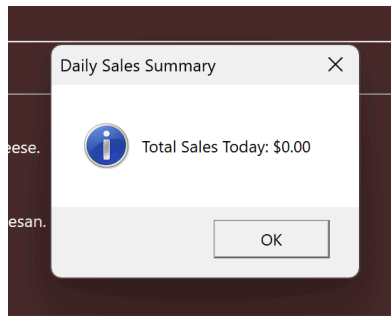


This is the AddItem Window when you press on an Add Button from the manager's window, It allows you to add a dish and save it. EXTRA: For image path, it has a list of option you are able to choose from for the picture of your dish.



The screenshot shows a window titled "AddItem" with a dark red background. At the top, there is a toolbar with various icons including a list, a camera, a magnifying glass, a square, a cursor, a refresh, a plus, and a checkmark. Below the toolbar, the title "Add Item" is centered in a large, bold, white font. Underneath the title, there are four labels on the left: "Name:", "Description:", "Price:", and "Image Path:". Each label is followed by a white input field. The "Image Path" field has a small downward arrow on its right side. At the bottom center of the window, there is a gray button labeled "Save".

This is a message box that tells the manager the daily sales summary. In this example, no orders were placed so the price is 0.



This is the food menu for any customers who want to order some food. You can add the menu items in your order and then place it or clear it if wanted.

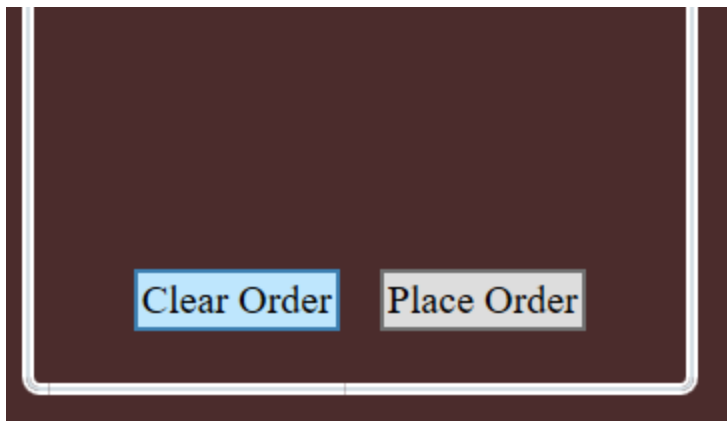


Adding items to the order will result in such appearance of your order section (example)

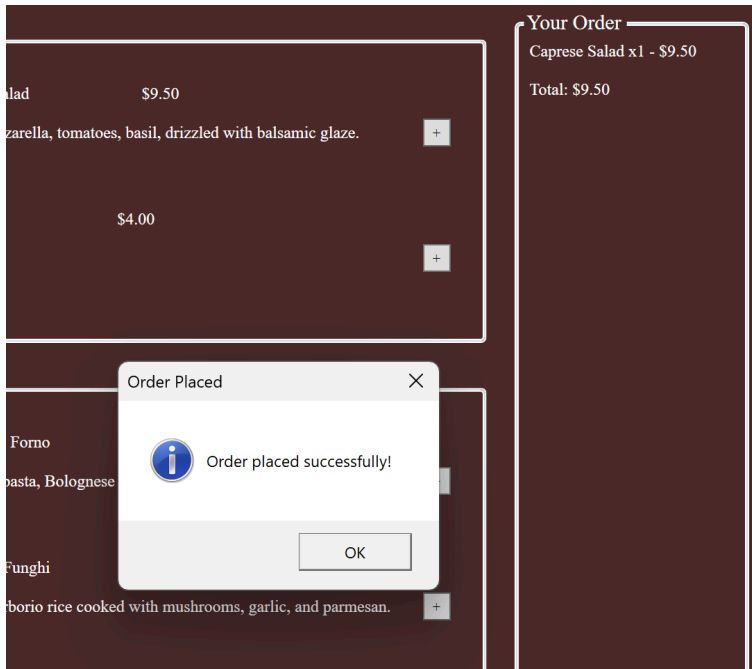


This is A Clear Order Button, after you add items in your cart, if you decide you don't want it anymore, it should clear up your cart and make it empty.

This is a Place Order Button, after you add dishes to your cart, you can choose to place an Order.



After you click place order, this message box will pop up to let you know your order was placed.



Future Work

Discuss features or improvement that can be added to application.

- Add an Order button in MenuPageForManager that allows you to see the order details from the customers, while seeing the total costs, the dishes selected and being able to delete them.

Appendix A: Team Contract

Submitted team contract goes here.

Team Contract Template

Sunday, December 4, 2022 10:36 PM

Appendix A: Team Contract Template

This is an informal contract to ensure that all team members have a common understanding of what is expected in terms of work standards, communication, division of work, and conflict resolution.

Team Members (Name & ID)

	Name	Student ID
Member A:	Antonina Kosyakova	2238996
Member B:	Alaa Kirdi	2363678

Strength & Weaknesses

Within the context of this project, what are the strengths and weaknesses that each member brings to the team?

Member A: My strengths in programming are my attention to detail and my perseverance in learning and improving. My weaknesses are avoiding advanced syntax like '=>' and spending too much time on small details, which can slow me down.

Member B: Strength: I prioritize writing precise and clean code, ensuring it is well-documented with comments for both my understanding and the readers. I also make a habit of thoroughly reviewing my code to eliminate redundancies, avoid magic numbers, and maintain a high standard of quality.
Weaknesses: I might need more detailed informations about the project and that might slow us down if I take too long trying to understand what to do exactly. Another is that I sometimes have poor time management because of all the projects due and additional work outside of school.

Definition of "good enough" for this project

What would the team collectively consider "good enough" of an achievement for the project?

(One response for the whole team)

A good enough achievement for your final project would be meeting all the core requirements, writing clear and organized code with proper comments, and including basic error handling. Successfully applying what you've learned is a solid accomplishment, and adding one small extra feature or polish can make your work stand out.

Picked Topic

Restaurant Menu

Division of work

How will each member contribute to the project?

Member A: I'll contribute by writing clear, organized code, implementing the main features, and helping with testing and debugging. I'll also work with my teammate to split tasks and finish the project on time.

Member B: Adding comments to clarify each step i took into the code, making the visual look nice, writing clean code and try to finish my parts on time.

Frequency of communication

How often will the team be in touch and what tools will be used to communicate? The team will stay in touch regularly through Teams private chat and social media, checking in as needed to share updates, ask questions, and coordinate tasks.

Response delays

What is a reasonable delay to reply to messages? Is it the same for weekdays and weekends?

A reasonable delay to reply to messages could be within a 2 hours on weekdays, unless we had a deal to work on the project on specific time slot (then it would be 15-20 minutes) and by the end of the day on weekends, depending on everyone's availability.

Receiving feedback

Each member must provide a sample sentence for how they would like to receive constructive feedback from their peers.

(If unsure, assume a hypothetical situation such as you have not completed your work in time or you have not replied to a message in a timely manner).

Member A: I would prefer constructive feedback to be given in a clear and respectful manner, focusing on specific actions or areas for improvement. For example, if I missed a deadline or didn't reply promptly, please point it out directly, explain how it impacted the project, and offer suggestions for how I can improve next time.

Member B: If ever I miss a deadline for one of my parts, I w

In case of conflict

If a team member fails to communicate as described in this contract or does not respond to constructive feedback, what measures should the other teammate take?

Message the other teammate until they respond and do their part, if that doesn't work then it is better to involve the teacher.

(One response for the whole team)

Appendix B: UML Class Diagram

- Do not include WPF created classes in the class diagram.

Type of Relationship	Classes	Reason
Composition	MenuManager → MenuItem	MenuItem is a part of MenuManager. Their lifecycles are tied.
Composition	Order → OrderItem	OrderItem is part of Order. Their lifecycles are tied.
Association	OrderItem * MenuItem	OrderItem references an independent MenuItem.
Dependency	RelayCommand → MenuManager	RelayCommand temporarily uses methods or data from MenuManager.

