Protocol for Tour Planner

Technical Steps

Project Setup

We created a Git repository and split the tasks between the two of us. I was responsible for the general setup and project structure, while my partner focused on the user-facing windows and input logic. The app is built using WPF and MVVM, and we decided early on to split the code into separate projects: the WPF app itself, a Business Layer, a Data Access Layer, and a separate project just for the data model. Each part was added to the Visual Studio solution and linked properly with project references.

The MainWindow acts as the starting point of the app. It contains buttons that let users create or delete a tour. Pressing these buttons opens pop-up windows where users can enter the necessary information.

Tour Creation and Deletion System

Clicking the "Create" button opens a window with fields for the tour name, start and end location, description, and transport type. When the form is submitted, the data is passed into a ViewModel and then sent through the Business Layer to the database via Entity Framework. The deletion system works the same way, but only requires the name of the tour to delete it from the database, we will probably add a direct deletion button to Tours when we add the Logic to View a Tour later on. All data is stored in a PostgreSQL database running in a Docker container. We use EF Core as the O/R-mapper and connected it to the database using a custom DbContext class.

Data Binding

We used the MVVM pattern throughout the project. The user input fields are bound to a Tour object in the ViewModel using standard WPF data binding. We also use a Relay class that implements ICommand to handle logic when buttons are clicked. This setup makes it easy to connect UI actions to code without writing logic directly in the code-behind.

At first, we tried to add placeholder text directly inside the textboxes, but that interfered with the bindings and caused issues. Instead, we decided to use labels above each field to show users what to enter. This approach was much simpler and doesn’t break the binding.

Logging and Unit Testing

We added logging using log4net to track everything that happens during runtime. Logs are currently not being written to the console but to a seperate .log file. I had to work through a few dependency and path issues to get this working right, but it now logs useful info like when a tour is created or deleted, and when errors happen.

We also added four unit tests using NUnit. Two of the tests check that clicking the “Create” and “Delete” buttons correctly open their respective windows. The other two test that the commands in the ViewModel actually run and perform the correct logic when a tour is created or deleted. We chose these tests because they check if the core user interactions are working — if those fail, the app can't do what it's supposed to.

Sadly we didnt have enough time to create a uml diagram or to show the Ui using wireframes.

Tracked Time

Date	Time	TeamMember	Task
17.02	1h	Both	Project Specifications Research
17.02	30 min	Sura	Set up GitRepo
17.02	1 h	Both	Split up tasks for Members
18.02	3h	Sura	Set up logical Project Architecture
24.02	3h	Sura	Set up the Main window to create tours
24.02	1h	Finn	Setup User Control for Tour Creation-
24.02	30min	Finn	Bound the main window to the popup window and the pop-up window to the user control

22.03	3h	Both	Looked into ViewModel
22.03	1h	Finn	finished Tour creation logic
22.03	1h	Finn	made same logic for tour deletion
22.03	3h	Finn	Made TourViewModel to receive and handle data received by View
23.03	30min	Finn	added relay commands
23.03	1h	Finn	made sure the pop up windows closed after they r no longer needed
23.03	1h	Finn	added labels to the textboxes to avoid confusion
23.03	2h	Sura	Included logging
23.03	30min	Sura	Added debugg logging
23.03	1h	Sura	Integrated or mapping
23.03	1h	Sura	Established database connection
23.03	30min	Sura	Created Business Logic for Tour Creation and Delete
23.03	1h	Sura	added Database methods for inserting or dropping tours

23.03	30min	Sura	integrated Unit testing dependencies
23.03	1.5h	Sura	added 4 unit tests