Tour Planner Documentation

App Architecture

In our project we implemented the MVVM Pattern, which by design implements these 3 Layers:

Presentation Layer, Business Layer and Data Access Layer.

In the Presentation Layer reside our Views, which present the UI to the user.

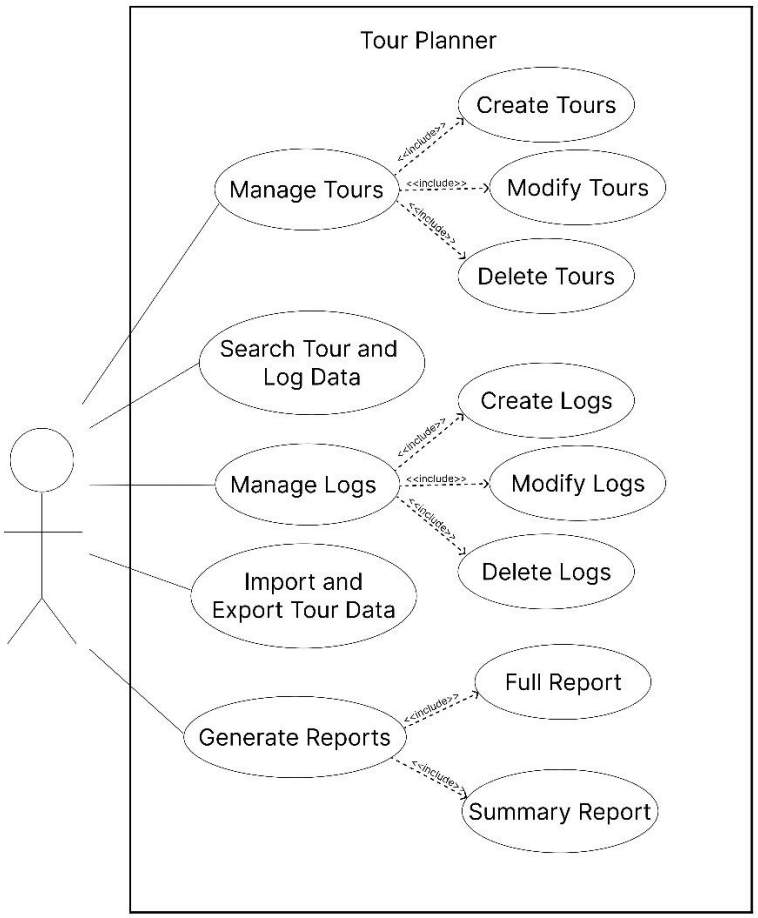
In the Business Layer you will not only find the ViewModels but some of the Business Logic is also handled by Utility Classes such as the ReportGenerator.

The Data Access Layer consists of our Models and the Database access.

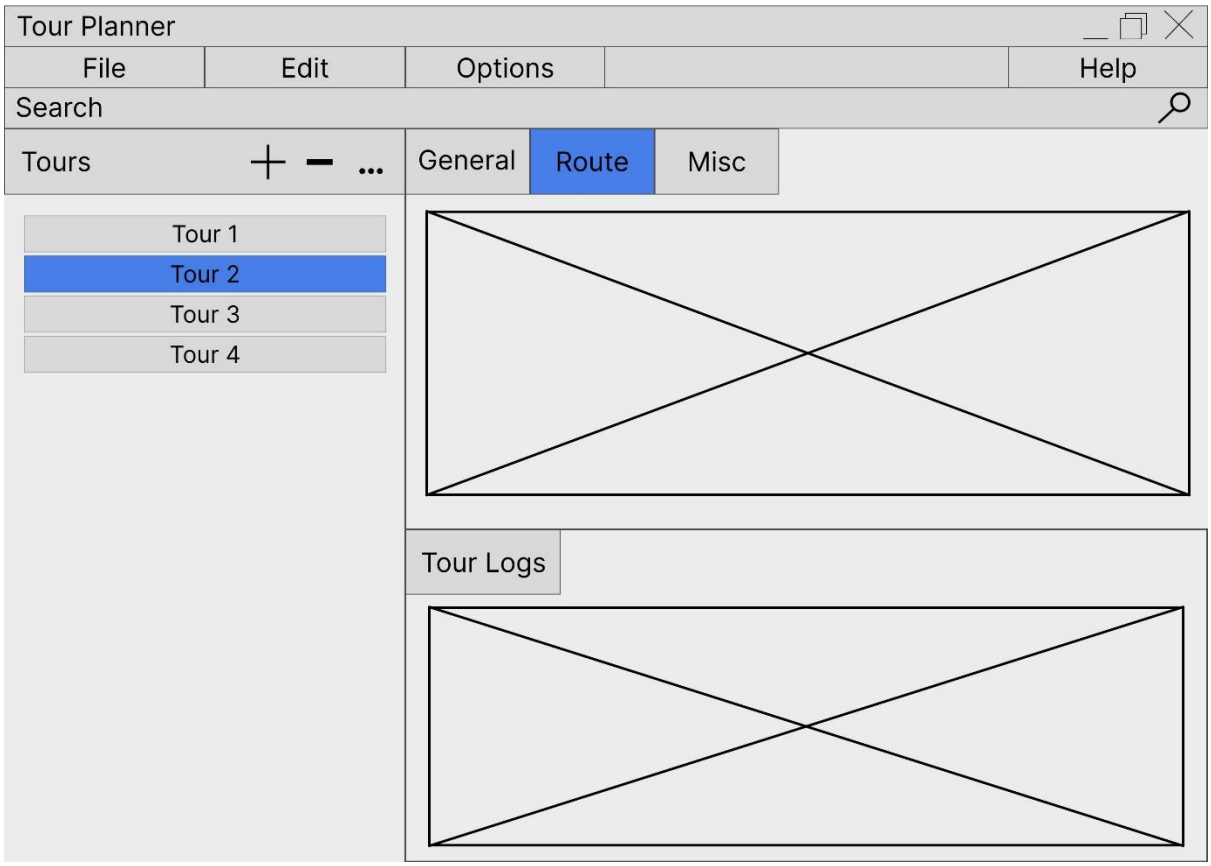
The Presentation Layer only gives commands to the Business Layer, while the Business Layer only gives commands to the Data Access Layer.

Use Cases and Wireframe

In this following Use Case Diagram, you can see the Use Cases for our application.



In the following Wireframe you can see how we planned the User Interface before we implemented it



UX

For this application, since we had no prior experience with WPF, we went for a simple style, which resulted in a clear, but not exciting, User Experience.

Design Patterns

The Design Pattern I want to highlight in our application is the Single Responsibility principle. One example of its implementation is the way we handle the configuration file. The configuration file holds critical information. Then we have a IConfig interface, which knows which variables are stored in the App.config file. Then we have the Config class, which retrieves the Settings from the App.config. This way, our classes don't have to access the configuration file themselves, but task the Config class with doing so. This is just one example of the Singe Responsibility principle in our application. Another example would be the InputValidator Class or the PathHelper. Both implement simple functionalities, which gets reused by other classes.

Unit Testing

In our project we have exactly 20 unit tests. That is because it was quite hard to come up with unit tests, because the program is not that big. We made one unit test for the DatabaseHandler using Mocking. This was quite the challenge, which is why I resorted back to unit test the rest of the application more thoroughly. In the future I would love to implement more Unit test which focus the Data Access Layer.

Unique Feature

Because of time constraints we have decided not to implement a unique feature.

Tracked Time

In total this project took us about 75 hours.

Lessons learned

One big takeaway of this project is how useful Interfaces can be. For example, I used an interface for the Config so that I can mock a config file for unit tests.

Another big lesson learned is that it is really satisfying to write code with good architecture. Although our architecture is far from perfect, this was the first time we were forced to try to implement a nice architecture and thus we could take away a lot from it and benefit from it in our future projects.

Link to GIT

https://github.com/straussdave/TourPlanner1