Meeting Planner

As a single student, develop a desktop application based on the GUI frameworks C# / WPF or Java / JavaFX. The user manages meetings in advance.

Requirements

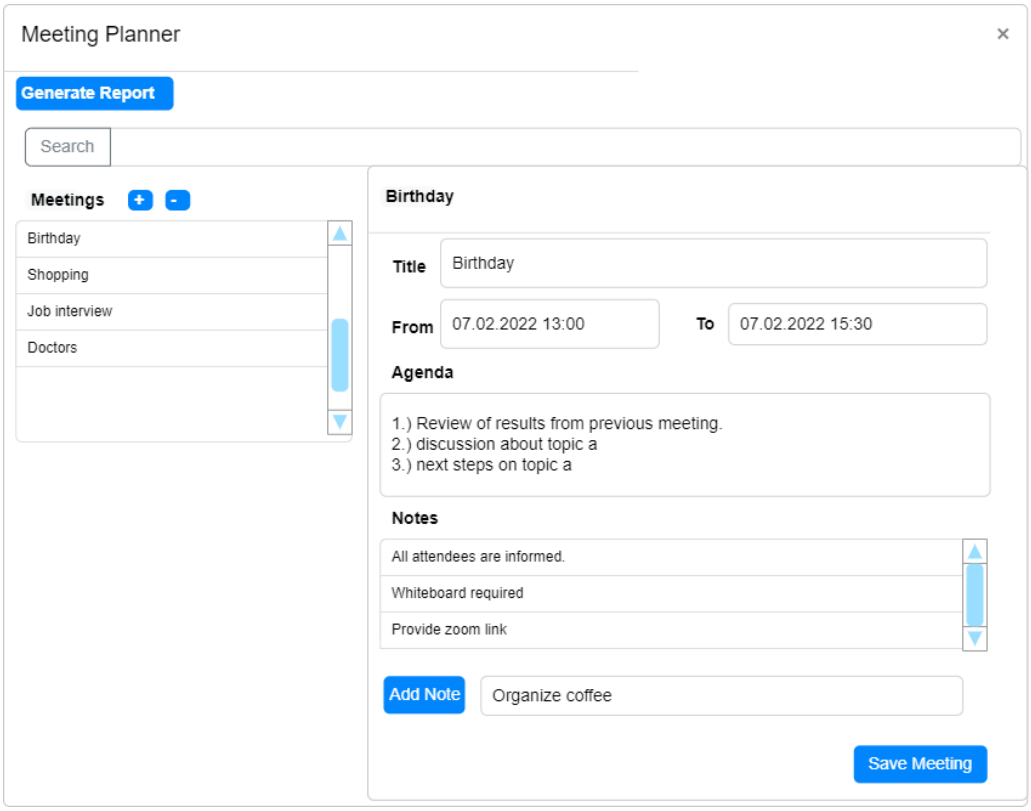
Goals

- implement a **graphical-user-interface** based on WPF or JavaFX
- apply the **MVVM-pattern** in C# / **Presentation-Model** in Java
- implement a **layer-based architecture** with a UI Layer, a business layer (BL), and a data access layer (DAL)
- implement **design-patterns** in your project
- persist the meeting-data and meeting-notes so that they aren't lost between application (re-)starts
- use a **logging** framework like log4net or log4j
- generate a pdf **report** by using an appropriate library of your choice
- generate your own **unit-tests** with JUnit or NUnit
- keep your **configuration** in a separate config-file - not in the compiled source code
- **document** your application architecture and structure using UML

Features

- the user can create new **meetings** (no user management, login, registration... everybody sees all meetings)
- every meeting consists of **title, from, to, agenda (simple text property)**
  - the user provides all the data in the GUI via appropriate input fields
- **meetings** are managed in a list, and can be **created, modified, deleted** (CRUD)
- for every meeting the user can create new **meeting notes**
  - multiple meeting notes are assigned to one meeting
  - a meeting note consists of a **text property**
- **meeting notes** are managed in a list, and can be **created, modified, deleted** (CRUD)
- **full-text search** in meeting- and meeting-notes data
- the user can generate a **meeting-report** which contains all information of a single meeting and all its associated **meeting notes**

(Rough) User-Interface Structure



Hand-In

Create a desktop application in C# (WPF) or Java (JavaFX) which fulfills the requirements stated in this document. Add unit tests (10+) to verify your application code. Upload your final code snapshot.

Add a protocol as pdf with the following content:

- document the application architecture using UML:
  - layer architecture + class diagram
- document the implemented design pattern(s)
- explain why these unit tests are chosen and why the tested code is critical
- consider that the git-history is part of the documentation (no need to copy it into the protocol)

For the final presentation prepare the following:

- present the working solution with all aspects
- execute the unit-tests and explain the results
- present the key items of your protocol (see above)

Mandatory Technologies

- C# / Java as desktop application
- GUI-framework WPF (for C#) or JavaFX (for Java) or another Markup-Language-based UI Framework
- Logging with log4j (Java) or log4net (C#) or another .NET Microsoft.Extensions-Solution.
- A report-generation library of your choice
- NUnit / Junit
- OR-Mapper (like .Net/Entity-Framework for C# or Java/JPA+Hibernate via Spring Boot for Java)

Grading

For a detailed point distribution see the accompanying checklist.

Must Haves

In case you don't implement the following required minimum goals, the hand-in is graded with 0 points:

- use a UI technology based on markup language (XAML, FXML)
- implement MVVM for the UI
- implement a layer-based architecture
- implement at least one design pattern (and mention it in the protocol)
- persist your data in some way
- store your application configuration in a config file
- integrate log4j/log4net for logging
- integrate a PDF generation library
- implement at least 10 unit tests

Points Distribution (100 Points)

- 35: functional requirements
  - GUI in general
    - design and function
  - meeting
    - create/modify/delete a meeting
    - view/manage meetings in a list
  - meeting-notes
    - create/modify/delete meeting-notes assigned to a meeting
    - view/manage meeting-notes as list
  - full-text search
  - generate report
- 15: non-functional requirements
  - persistence
  - configuration

    - unit-tests
- 10: protocol
    - design and architecture
    - unit test design
    - link to git
- 10: presentation
- 30: theory