Tour Planner Project Checklist

## 1. Project Setup

|  |  |
| --- | --- |
| **Task** | **Done ✓** |
| Team of 2 students formed | ✓ |
| Technology stack selected: C# with WPF or Java with JavaFX | ✓ |
| Version control initialized (Git recommended) | ✓ |
| Time tracking method decided | ☐ |
| Configuration file created (stores DB connection, base directory) | ✓ 🡪 base directory stored but not used |

## 2. Architecture and Design

|  |  |
| --- | --- |
| **Task** | **Done ✓** |
| Layered architecture: UI, Business Layer (BL), Data Access Layer (DAL) | ✓ |
| MVVM (C#) or Presentation-Model (Java) pattern applied | ✓ |
| At least one design pattern implemented | ✓ |
| Custom reusable UI component created | ✓ |
| Application configuration externalized (not hardcoded) | ✓ |

## 3. External Integration

|  |  |
| --- | --- |
| **Task** | **Done ✓** |
| PostgreSQL database setup via O/R-mapper | ✓ |
| OpenRouteService + OpenStreetMap APIs integrated | ☐ |
| Logging framework integrated (log4net or log4j) | ☐ |
| PDF generation library integrated | ☐ |

## 4. Functional Features - Tour Management

|  |  |
| --- | --- |
| **Task** | **Done ✓** |
| Create new tours (Name, Description, From, To, Transport Type) | ✓ |
| Automatically fetch Distance, Estimated Time, Route Image from API | ☐ |
| Display and manage list of tours (View, Edit, Delete) | ✓ |
| Input validation implemented | ☐ 🡪 not entirely |

## 4. Functional Features - Tour Logs

|  |  |
| --- | --- |
| **Task** | **Done ✓** |
| Create, view, edit, delete tour logs per tour | ✓ |
| Tour log includes: Date/Time, Comment, Difficulty, Distance, Time, Rating | ✓ |
| List of tour logs shown and managed | ✓ |

## 4. Functional Features - Computed Attributes

|  |  |
| --- | --- |
| **Task** | **Done ✓** |
| Popularity (based on number of logs) | ☐ |
| Child-friendliness (based on difficulty, time, distance) | ☐ |

## 4. Functional Features - Search and Reports

|  |  |
| --- | --- |
| **Task** | **Done ✓** |
| Full-text search on tours and logs (including computed fields) | ☐ |
| Import/export functionality for tour data | ☐ |
| Unique feature added (custom functionality of your choice) | ☐ |
| Generate single Tour Report with all logs | ☐ |
| Generate Summary Report with average time, distance, rating | ☐ |

## 5. Testing

|  |  |
| --- | --- |
| **Task** | **Done ✓** |
| At least 20 unit tests written using NUnit or JUnit | ☐ |
| Key logic covered (e.g., data persistence, computed values, API interaction) | ☐ |

## 6. Documentation

|  |  |
| --- | --- |
| **Task** | **Done ✓** |
| Protocol PDF written | ☐ |
| Technical decisions and challenges explained | ☐ |
| UML Use Case Diagram | ☐ |
| UI Flow Wireframes | ☐ |
| Class Diagram | ☐ |
| Sequence Diagram (Full-text Search) | ☐ |
| Justification of test cases | ☐ |
| Time tracking summary | ☐ |
| GitHub/GitLab repository includes full commit history | ☐ |

## 7. Final Presentation

|  |  |
| --- | --- |
| **Task** | **Done ✓** |
| Demonstration of full working solution | ☐ |
| Execute and explain unit tests | ☐ |
| Present highlights of protocol (design, architecture, testing) | ☐ |

## 8. Bonus (Optional)

|  |  |
| --- | --- |
| **Task** | **Done ✓** |
| REST server for data management | ☐ |
| UI synchronization for multi-user environment | ☐ |
| Conflict prevention for concurrent edits | ☐ |

27% through