Checklist Tourplanner Exam - Final Hand-In

Review Date:
Student Name:
Personal Identifier:

Must Haves

	Yes(1)/No(0)
Uses C# or Java	1
Uses markup-Based UI framework	1
Uses MVVM for UI	1
Implements a layer-based architecture (UI/BL/DAL)	1
Implements at least one design pattern	1
Uses a Postgres Database for storing Tour Data	1
Does not allow for SQL injection	1
Uses an OR-Mapping Library	1
Uses a config file that stores at minimum the DB connection string	1
Integrates the OpenRouteServices.org and OpenStreetMap APIs	1
Integrates log4j/log4net or similar Log Libraries	1
Integrates a report-generation library	1
Implements at least 20 Unit Tests	1

Features

	Points
GUI in general	
Correct data binding between UI elements and view model properties	
UI responds to window size changes	
Defines reusable UI Component	
Tours	
Create/modify/delete tour (also in DAL)	
Tours have required attributes (incl. Image) and are managed in a list view	
Tours have computed attributes	
Tour Details show all tour attributes of a selected tour and also the map image	
Validates user-input (no crash on wrong input)	
Tour Logs	
Create/modify/delete tour log (also in DAL)	
Tour log has required attributes	
Tour Logs showing all logs of a selected tour with all log attributes in a list view	
Validates user-input (no crash on wrong input)	
Full-Text Search	
Search performs full-text search in Tours, Tour Logs and computed attributes	
List of Tours according to current search	
Reports, Import/Export	
Single tour report (with Map Image)	

Summarize report
Export tour data
Import tour data

Mandatory Unique feature

Non-Functional Requirements

Layers only call methods of the immediate layer below (or own methods)
Layers define their own exceptions, no implementation specific exceptions
Uses the OpenRouteServices.org Directions API for tour retrieval
Uses an OpenStreetMap Tile Server for the map
All tour data (maybe except the image data) is stored in the database
All configuration information is stored in a configuration file
Logs exceptions, errors and other useful technical information
Quality of unit-tests (usefulness, no duplicates, ...)

Protocol

Describes app architecture (layers, layer contents/functionality, class diagrams)
Describes use cases (include use-case and sequence diagrams)
Describes UX (include wireframes)
Describes library decisions (where applicable), lessons learned
Describes implemented design pattern
Describes unit testing decisions
Describes unique feature
Contains tracked time
Contains link to GIT

Bonus Features

Sum Points 0

Max. Points
1 2 1
1 1
2 2
2

1
2
2
2
2
1
2
2
2
1
1
1
3
1
1
1
1
1
0.5
0.5

40

Comments

For Java: The BL and DAL must use Spring Boot. For C#: Use Entity Framework or equivalent.

Comments