"Tourist" Project (Software Quality)

The chosen Software Quality Model is the ISO/IEC 9126 Model. The application is one for mobile devices (Windows Phone only for now) that can be used to organize touristic attractions. Each user has a personal list of attractions that he/she wishes to visit. In order to add attractions to this list he/she must search for them and then from each attraction detailed view to add them to the list.

The detailed view includes a consistent description, comments, rating (from 1 to 5 where higher means more attractive), pictures and the possibility to view the attraction on a map. Only users that have visited the attraction can rate and comment.

The quality factors that will be traces are as follows:

Usability

In order to evaluate usability there is need for a number of users (at least five). Each user gets a mobile device with the application during the usability evaluation and asked to perform tasks that the application offers. The one gathering user feedback will observe how the application is used in order to determine how intuitive the user interface is. After performing those tasks the user will take a survey rating features on a scale from 1 to 5.

Efficiency

To determine efficiency each feature will be timed a number of times and then compute an average. The average will tell whether the application is responsive enough (e.g. it does not take minutes to perform a simple task).

Maintainability

Maintainability is strictly code related. The code will be measured using tools that calculate the Maintainability Index and Cyclomatic complexity.

Results

Usability

After evaluating a few users it was noticed that most issues they had with the interface were common. The issues/proposals are listed below.

- When the application starts it is not clear what the initial list is mean for, some users thought that the application has no attractions recorded.
- Another issue was with the “pin” or “add to my attractions” button, because it offers no feedback to the user. When it is pressed it is unclear what the button does or if it did anything.
- The pivot navigation (tab-like) is a bit counter intuitive when swiping from details to comments and then to map when viewing a specific attraction.

- As users propose pictures for attractions that are already posted, they should be approved by the user which published the respective attraction otherwise ill intended users would be able to post unrelated or offensive images.
- When viewing pictures, it would be more comfortable to swipe from one picture to the next without having to go back to the picture list view and then selecting the next picture. When having a large list it could be problematic as the user would have to identify which was the previous picture in order to avoid seeing the same ones twice.
- The buttons for rating and pinning attractions are a bit confusing, the icons are not intuitive enough which may force users to expand the command bar in order to know what the buttons do.
- Last but not least, a user should have privacy settings for his/hers personal attractions list in order to allow or forbid users from seeing it.

Overall the application is intuitive enough however it could use some final touches to improve the user experience and remove the minor ambiguity.

Efficiency

Each feature was timed separately for five times and then each time was used to compute an average. Below is a table containing the average times for each feature.

Feature	Average time
Loading attractions to visit	00:00:00.4683097
Searching for attractions	00:00:01.1141502
Select attraction	00:00:00.2742432
Add attraction to "to visit" list	00:00:00.2172171
Loading comments	00:00:00.6600771
Add comment to attraction	00:00:00.2056114
Loading attraction details	00:00:00.1017134
Adding an attraction	00:00:05.2250178
Viewing attraction on map	00:00:00.0321611
Viewing nearby attractions	00:00:06.0264221

One can notice that most tasks are performed under a second. The most time consuming tasks use the GPS device on the phone which requires going through the Windows Phone OS and then triangulating with satellites (time consuming operations not necessarily complicated ones). Searching is fast enough however the search time will grow as more attractions are added.

Overall the application performs fast enough however periodical checks on how fast the search turns out to be should be made in order to maintain user satisfaction.

Maintainability

Looking at the results in the *QA Project Code Metrics Results.xlsx* file it can be noticed that highest Cyclomatic complexity is 19 and right after that is 18, 17 and 15. Afterwards there are a few of 9s, 7s, 6s, 5s and 4s which are somewhat alright but should be refactored. Then is a long stream of methods having the Cyclomatic complexity between 1 and 3 meaning the respective methods have up to 3 possible execution paths. That is easy enough to understand.

Looking at the Maintainability Index it is somewhat proportional to the Cyclomatic complexity. The method having 19 possible execution paths has the lowest Maintainability Index telling for a second time that it should be refactored. That method is trouble. Next “troubling” methods are the same ones indicated by the Cyclomatic complexity. Then a part of the results are mixed showing that even methods with one execution path need refactoring as they can be quite long and hard to understand.

Overall with a few exceptions the application is maintainable and can be extended quite easily.