Peer Review for Roy Nilsson, Oskar Klintrot and Mikael Eriksson

Test of the runnable application.
Works good, no bugs noted! Have added members, boats, listed them, updated them etc. All operations works as intended.

Test of compiling the project.
Compiling the project worked almost without problems. We didn't have .NET 4.5.2 installed on our machine but changing the target-version to 4.5 did not blow up the whole application!

Implementation vs Diagram
Since the diagram is reverse-engineered in VS it conforms with the implementation. The sequence diagrams give a good overview of what's going on.

Is the architecture ok?
Overall we think the architecture is good. There is a clear separation of the logic and the UI-layer via the model view separation. It is also good that you have done it with MVC, a controller that manages the flow of the application, it makes the code easier to read.

Perhaps the cohesion could be higher [1, p432, ch17.8], with more controllers instead of one ApplicationController that handles everything. A good idea would be to have one controller that handles Member-operations and one that handles Boat-operations.

We think that the BoatView shouldn't call services of BoatDAL directly, this is a job for the controller. If this would be a strict layered arcitecture, this would be forbidden, as Larman states [1, p317 ch13.2]. In a relaxed layered architecture this is OK for a higher layer to call upon services not directly under them.

Is the requirement of a unique member id correctly done?
Yes, you get the hightest ID and increment it by one.

Quality of the implementation/source code
The overall quality of the source-code is good. We like that you have used a string resource-file to eliminate string-dependancies. Good naming of methods that makes it easier to read the code.

What is the quality of the design? Is it object oriented?
The quality of the design is good. As mentioned earlier the cohesion could be higher [1, p432, ch17.8]

As a developer would the diagrams help you and why/why not?
The diagrams help us understand the code better. It gives a good overview of what classes interact with each other. This could be achieved by reading the code but looking at a diagram makes things easier.

What are the strong points of the design/implementation, what do you think is really good an why?
Enumerable for the listoptions is good. String resources as well. Dependancy injection of the views to the AppController.

What are the weaknesses of the design/implementation, what do you think should be changed and why?
As we mentioned before, the cohesion could be higher and the BoatView should not call BoatDAL directly. Should have used more exceptions instead of returning -1 from the BoatView of boats are null.

Do you think the design/implementation has passed the grade 2 criteria?
Absolutely!

References

Larman C., Applying UML and Patterns 3rd Ed.
Please note that the page numbers referenced in this document is from the PDF-version, not the hardback copy. Link to the pdf:
https://drive.google.com/file/d/0B3-OzxP8Wm1IbmhhTGhIMXlQYlU/view?usp=sharing