**Eng 73 : WPF-Entity Framework C# Project – November 2020**

**Start – Wednesday 04 November**

**Final Sprint Review – Tuesday 10th November 2 pm**

**Presentations –14:30 am Tuesday 10th November**

**Introduction**

See marking scheme at the end of this document for more detail.

Create an application! It must:

* Have an WPF front end
* Have an SQL database backend with at least two linked tables
* Use Entity Framework to manage the relationship between your backend object model and database
* Have a Business Layer with some logic - not just a simple CRUD application
  + The Business layer code should be covered in Unit tests, which exercise the normal functionality, boundary and error conditions.

**Project management**

Create a new Git-Hub repository for this assignment. It can be private, but you must invite me (mbeard88) to join it.

Create a Git-Hub Project board associated with the repository. It should have at least 6 columns:

* Project Backlog,
* Sprint Backlog,
* In Progress,
* Review,
* Done,
* and Notes.

The Notes column should start with 3 notes stating your Project Goal, Project Definition of Done, and generic User Story Definition of Done. See the “User Stories, Backlog and Estimation” lesson for examples.

Create cards representing epics and user stories and put them in priority order in the Project Backlog.

**At the start of each sprint**, define a sprint goal and put it (as a note) in the Doing column. Break down epics and flesh out user stories and put the stories to be implemented in this sprint into the Sprint Backlog.

Cards representing user stories should include:

* A brief title including the Sprint Number
* Description in the form AS a\_\_\_\_ I need \_\_\_\_\_ so that I can \_\_\_\_\_\_
* A checklist of acceptance criteria / scenarios that should be catered for

**At the end of each sprint**, following the Sprint Review, all accepted User Stories should be moved into the Done column, and the Note of the Sprint goal should be moved to the Notes column. User stories which are not complete / accepted should be moved back into the Project Backlog.

Take a screenshot of the Project board at the start of each sprint and end of each sprint.

**Commit** your work regularly to Git, using clear and descriptive commit messages which include the title of the associated User Story or Task.

**If you prefer**, you can use Trello\*\* instead of the Git Project board to store your Kanban board. Be sure to invite Cathy to it and ensure your Git commits are clearly associated with story cards.

**Documentation** – put this in your Git project, in an .md file

Be concise. Use bullet points and/or tables, not long blocks of text.

* Project goal and definition of done
* For each sprint:
  + Sprint goal
  + Output of sprint review: The list of backlog items "done" in this sprint, actions for any items not “done”
  + Sprint retrospective: A list of things that went well, improvements and action plan.
* Overall project retrospective – what have you learned, what would you do differently next time, what would you do next?
* A note of which class diagrams are included in your code projects – name of the diagram(s), and where to find them.
* Screenshots of the Kanban board (start and end of each sprint). These can be in the same file alongside the documentation of each sprint, or together in a separate document.

**Code**

I will mark the code last committed into your main branch by 2pm Tuesday

* Include in your Solution the Class diagram(s) showing the structure of your finished product.

**Group Presentation** with individual demos

See notes for more detail

**Timetable** (subject to events)

|  |  |
| --- | --- |
|  |  |
| Wednesday  *Project Planning and start of Sprint 1* | Morning – Project introduction,  Decide on overall project goal and definitions of done, and create project backlog (user stories should be high level at this point).  **Plan and start sprint 1**  From 2:30 – 1-2-1 where we will look at your Project/Trello board and discuss your project goal and backlog |
| Thursday *Sprint 1 (cont)* | 9.30 am group standup – show your Project/ Trello board  3:30 pm **End of sprint 1** Individual sprint review meeting with CLF (from 3:30). Individual sprint retrospective. **Plan and start sprint 2** |
| Friday *Sprint 2* | 9:30 Academy standup  10:00 am Group standup– show your Project/ Trello board. |
| Monday *Sprint 3* | 9.30 am **End of sprint 2 start of sprint 3**. Group sprint review meeting. Each of you will demo your work done so far and give each other feedback.  Individual retrospective.  **Plan and start final sprint.** |
| Tuesday 2 pm | 9:30 am Group standup– show your Project/ Trello board.  2:00 **End of final sprint** Final code to be checked into GitHub by 2:00 Group sprint review meeting, class discussion of presentation format  Individual retro and finalisation of documentation  **End of final sprint**  Prepare project presentations and have a whole group rehearsal |
| Wed am | 10 am Project Presentations |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Categories | Unskilled 1-2 | Low-Skill 3-4 | Partially Skilled 5-6 | Skilled 7-8 |
| Functionality and extensiveness  40% | Very little implemented successfully. No database or no GUI. | Not an extensive product, but some functionality implemented. May be missing a layer (GUI or database) or crash in some scenarios | Good product with reasonable functionality, featuring a GUI, logic layer and database. May have minor flaws, but for the most part is easy to use and prevents/handles invalid input and errors. | Extensive and robust functionality, with extensive GUI, rich logic layer, and at least two database tables. Has excellent usability and innovative features. |
| Unit tests  10% | Little or no unit testing | Some business layer functionality covered with tests, some may be badly implemented or not complete. | Most business layer functionality covered with tests, some omissions | Business layer functionality well covered with tests, including boundary values and error cases. |
| Code quality  20% | Code is not abstracted out in a way that makes it easy to reuse or maintain in any way. Classes, fields, properties and methods are not clearly named, indentation is poor or absent and logic is difficult to follow.  Poor logic, repetition, little use of OOP principles or database design. | Code may be properly indented and spaced-out, but the flow of logic is difficult to follow. Poor choice of class, field, property and method names.  Some use of classes and other OOP principles, but not well implemented. Code is repetitive or inefficient. Design of the database layer and its interaction with code is poor. | Logic is fairly easy to follow, but parts of the code take time to decipher. May not conform to naming conventions or be overly complicated.  Reasonable class and database design and implementation with some minor flaws. | Well designed code with good use of OO principles in an extensive implementation. Code is concise and clear. Logic is naturally easy to follow. All elements are named in a way that make it easy to understand their purpose. |
| Project management  20% | Disorganised approach with little documentation or use of Kanban board. Project is either not git tracked at all or has only one or two commits. | Kanban board is sparsely populated and/or documentation has gaps, scrum process not entirely followed. Project is git tracked, but not properly managed, with not many commits. | Kanban board is well utilised and documentation is nearly complete, showing structured approach using Scrum, with some adaptation and improvement over the week.  Project is tracked with at least one commit per user story. | Kanban board is very effectively used and documentation clear and concise, showing logical approach using scrum and with constant adaptation and improvement.  Project is well tracked with multiple commits, each with clear and descriptive commit messages. |
| Presentation 10% | Minimal presentation | Presentation shows some of the functionality, but there are major omissions, or the overall purpose and scope of the work is not clear. | Presentation reasonably describe the work done but may have minor omissions or timing issues. | Product is clearly presented, functionality well demonstrated in a logical order with appropriate choice of test data. |