SEM5640 Individual Report

# Personal Evaluation

Each point of contribution (what I did) needs to state:

* How I worked as part of the team?
* Critically evaluate (reflect on it) the good and bad of my work?
* How would I do it differently?
* Process
  + I was strongly pushing using a product and sprint backlog at the very start of the project because when learning agile I read that flaws in the backlog was one of the main signs of low discipline which results in scrum failing. I think this push was too hard instead the backlog should have developed more naturally. The backlog I suggested when breaking down the project into tasks was just useless, what exactly to include in the backlog was also unclear. We really needed a way to plan and track tasks but only one that would work for this project. Not something from a text book that was to be abandoned after 2 weeks. I made to excel spreadsheets one for the Product backlog and one for our sprint backlogs.
  + I suggested we record our end of week meetings (review/retrospectives) but this didn’t last more than a week. I did not commit to this enough although the original idea was good as a way to maintain a project review history. I think we didn’t really know what to do if the result of the review was “We didn’t finish the work” or “The work is currently in progress”. Maybe commiting to a honest cirtical review of the work completed so far would have been a better option than not reviewing at all.
  + I used gitlab for version control throughout the project and using it went well.
* Documentation
  + Component diagram was assigned to dom but redone by me (I think he was under confident with UML and very under confident in using visual paradigm)
  + I suggested we use a uniform formatting to make our documents all look the same I took this from the 2nd year project so it was no work on my part but the idea to do this was good because it makes docs more professional and easier to read. I think this was a good idea because it was an easy investment of effort given the format was already there, if doing the project again I would use this format again, if it was a serious project I would make the document nicer to reflect my own taste using latex but for a busy project the template used was easy and fit for purpose.
  + I started doing design I pushed for Entity relationship diagram and api design first because I think focusing on the data responsibilities of each service and how they exchange data is important in understanding the system. The ER diagrams were in the form of UML ORM diagrams without the object mapped classes half of the diagram. The API design was a table of REST endpoint URLs along with a description and status codes for each HTTP verb (GET, POST, PUT, DELETE)
  + I researched Solr and wrote the Solr Technical report deliverable. There was not a lot of research because solr advertises its benefits and popularity quite obviously. Most of my reading into solr was time spent playing with it to get a feel for the technology rather than doing proper research. It was clear from the offset of its relevance & usefulness to our project.
  + I spent a bit of time reading into example test strategies for small scrum projects I shared the resources I found with dominic because he was going to write the Test Plan deliverable.
  + I drew a use case diagram for the project in Visual Paradigm (We used a single visual paradigm project .vpp for all our project UML) the diagram was made by both me and dom contributing at a single computer I was just drawing it because of my familiarity with the Visual Paradigim modelling software.
  + I wrote the Project Plan Deliverable.
  + I added the mapping class half to the ORM diagrams towards the end of the project
  + It was always my intention to produce class diagrams for each microservice I started
* Testing
  + Jmx tests for notification, message store and module registration
  + Unit tests for module registration (staff/student/module controller) unfnisyhed file controller unit test
  + Used CURL in bash script to test CSV upload endpoint because we had trouble getting those to work with JMeter
  + Used Moq In unit tests
  + Tried to get junit test working but couldn’t get embeddable EJB container into unit tests which is required for any meaningful tests, therefore no unit tests for message store.
* Deployment??
  + Added Dockerfile and runner script in each project I was responsible for. This script would take down, rebuild and start the project.
* Message Store
  + Logging in Java
  + JAX-RS
  + Eclipselink moxy for JSON
  + JPA
  + Time Learning Unit tests (but no luck with emdebbed EJB container)
  + Use of netbeans with glassfish and PostgreSQL connection
  + Added a search controller to Message Store. JAXRS endopoint to POSt key value apris required for search. Then used solrj code to index and search the data with solr. Returned the result as custom json object.
* Notifications
  + Quartz jobs
  + Setting CRUD with REST actions & entity framework core
  + Refactored code to use IDataRepository as an abstraction layer to accessing the database.
  + Added email service using SmtpClient with some HTML formatting to make the emails look a little better. The email formatting of was very basic with not all message data included. The plan was to revisit it and design a better looking email format but I didn’t have time to do that because of other priorities and lack of time during the last weeks of the project.
* Module Registration
  + Dom was assigned to write module registration but there were no foreign keys and problems with it to the point where it was completely unfunctional although the logic he had written for the CSV upload and parsing endpoint did mostly work.
  + I rewrote the module registration to have alternate models that linked to a database using surrogate auto id primary keys.
  + I changed the code to have common conventions in naming, types used and comments.
  + I abstracted the db context access through use of an IDatareposioty as recommended in the practicals.
  + I fixed bugs in the controllers I found with my jmx tests.
  + I changed return status codes and url routes to be more sutiable.
  + I made controller and data repository access operations asynchronous.
  + I added logging to the file controller. Although no sure this was needed.
  + I refactored and debugged the CVS upload file controller actions.
* Frontend
  + <TODO>

## Process Contribution

At the start of the project I suggested we use sprint and product backlogs to track our tasks. I put together two Excel <REF> documents one for the Product Backlog and one for our weekly Sprint Backlogs. I was strongly pushing for using these documents because I know they are a key part of Scrum <REF> and I had read that failing to maintain a proper backlog was a serious sign of a lack of discipline in a Scrum project which often results in Scrum failing. I think my initial intention to be committed to as close a scrum-like process as possible was a good thing. Dom agreed to this but did not contribute to the design of the backlog. This backlog was not used after the first two weeks because it was done with uncertainty and the team did not engage with it. How to break down tasks and what exactly to include in the backlog was unclear. I think we really needed a way to plan and track tasks but only something that would work for this team and this project; not something from a text book that was to be abandoned after two weeks.

I suggested we record our end of week meetings. These were intended to be a combination of Scrum’s Sprint Review and Sprint Retrospective rolled into one. I drew up the format for these minutes but their use did not last more than a week. When recording the meetings both of us gave ideas and I wrote them down in my tabular format which was part of a dynamic Word <REF> document. I did not commit to this enough although the original idea would have been a good way to maintain a project review history. These were abandoned because the team found no value in them. I think we didn’t really know what to do if the result of the review was “We didn’t finish the work” or “The work is currently in progress”. Maybe committing to an honest critical review of the work completed so far would have been a better option than not reviewing at all.

Throughout the project I used Gitlab for version control and my use of it was successful. I think using branches to always keep a working master would be an improvement. I suggested we use different Gitlab group projects for each microservice which worked well for both of us.

## Documentation Contribution

I suggested we use a uniform formatting to make our deliverable documents more professional and easier to read. I took the template for this format from the 2nd year group project module so it was no work on the team’s part to create the format. This idea was a good contribution because it was an easy solution to documentation that required no investment of effort with the resulting template easy to use and fit for purpose. If doing the project again I would use this format but perhaps make the document visually nicer using latex to reflect my own taste.

I wrote the Project Plan deliverable. This document was unfinished and a poor piece of work. It identifies a few good things but does not expand enough. I think it was supposed to be revisited but got forgotten as other things took priority. Taking this document seriously would be an improvement if redoing the project. – remove this its just negative & WORD COUNT???

## Testing Contribution

## Implementation Contribution

# Group Evaluation

How well group worked?

I felt I had all the work to do and no one to help me. Dom did not understand things properly which is fine because we here to learn but the problem is he took no care or want to learn for himself. Everything he did had to be redone by me. I feel I was stretched too thin because I couldn’t properly focus on one thing given. I had to fix his work or jump prematurely between different things. It was tiring because I had to initiate every pro-active thought or decision.

Dom would not draw design UML or do any testing of his own back. I introduced JMeter as a way to test endpoints and he did take that on properly but that is all and only after pushed him to do so. I did very little testing myself I really wanted to focus on it from the start but delays in the project and no encouragement or help from my group on the testing part meant any testing problems I had were always put on the back burner.

Managed working with docker well. Dom had a knack for it learnt things of his own back and showed me some stuff. Direct connections of our local running code to the actual databases meant we could all check what was going on for each other and in a way being able to inspect the database your team member is modifying, in my opinion, made us more of a team

How successful was the development methodology? Would use dev method again?

Evaluate group performance and output? What could be improved?