Group Project – Interim Group Report

Client: Dr F C Langbein

Product

Owner: C Allen

Team 1: R Watson

A Lamnea

W Cooter

S Tomlinson

Team 2: Ellis Doran

E Joiner

H Nicholson

J Davies

Back End: Ethan Kelly

Group 10 was broken into three primary work groups. Team 1 and team 2. I broke it down like this so that the work could be shared out with more structure. This allowed the authority to be shared too by having two team leads. I felt this was the best approach initially as it allowed the task to be broken down into a series of manageable tasks which could be concurrently developed.

Following our initial meeting with the client we a number of requirements that we broke into functional and non-functional requirements.

|  |  |
| --- | --- |
| Functional | Non-functional |
| Upload proposals | Flexible on group members |
| Assign students to groups | Must function on minimum input |
| Manually add/remove students | Must be easy to add to |
| Provide mark moderation | Must present a username |
| Must ensure simultaneous release of marks | Must include tools to help students complete group project |
| Mark deadlines |  |
| Give students tools to manage projects |  |
| Allow for pseudo-random assignment to groups |  |

In order to meet these criteria we broke the system into 4 major parts which should work together to various degrees in order to successfully meet all requirements. These are a file upload system, a grouping tool, project tools and a grading tool.

We started development of the file upload system to begin with, this allowed us to produce a core system, which could be added to later in order to allow mark moderation and a grouping tool together. The file upload system was given to team 1 in order to provide a working foundation for all the code we would produce.

To guide Team 1 through the development process I generated some testable requirements to meet throughout the design. These were developed from the requirements given by the client and from issues we identified in the planning process.

To speed up development it was my choice to develop a python script to produce a list of students that are formed into groups. This allowed the development of the most important and the hardest to implement tools first. Following these two tools development my plan was to present to the client the option to have both of these tools separately working or whether to integrate them.

Team 1

[Delete everything between square brackets when done, these are a content guide and should help the report follow the same themes require for your part of the report]

[Short intro:

-Requirements

-How you expected to meet them]

[2-4 para

-How you planned your data structure use (why you used array instead of map or w/e)

-Talk about cohesive code design

-Talk about how it is coupled with other elements of the program

-Explain how it solved the problem]

[UML of your part of the system (speak to Chloe if you need help)]

[3-5 paragraphs

-Explain how the testable criteria was met

-Explain issues you have encountered

-Talk about how you think these issues could be solved]

[Summary

-Explain why the structure of your program is the best within the requirements

-Talk about team level decisions you would redo]

Team 2

[Short intro:

-Requirements

-How you expected to meet them]

[2-4 para

-How you planned your data structure use (why you used array instead of map or w/e)

-Talk about cohesive code design

-Talk about how it is coupled with other elements of the program

-Explain how it solved the problem]

[UML of your part of the system (speak to Chloe if you need help)]

[3-5 paragraphs

-Explain how the testable criteria was met

-Explain issues you have encountered

-Talk about how you think these issues could be solved]

[Summary

-Explain why the structure of your program is the best within the requirements

-Talk about team level decisions you would redo]

Back-End

The back-end of the project refers to the organisation of the group’s GitHub, resolving coding issues and the construction of the database for the final system

One issue we noticed during the development of our project, was that the file upload system and grouping tool handled data in two very different ways. The file upload system used a file directory which was only accessible to the PHP files used to make the website and provide functionality to the system. The grouping tool used a CSV file which would load in the appropriate fields related to students, and then after grouping students, the updated data would be exported to a new CSV file.

In order for these two systems to work together efficiently on the same website, we need to centralise the data into a database. This database will store data for the whole project. Relevant data will be retrieved/updated/deleted when the corresponding part of the website invokes the database.

The database is suitable for the this task, as it can be queried; instead of searching through files stored on a server somewhere or fields in a CSV file, by querying the database with an SQL statement such as:

SELECT \*

FROM Logins

WHERE Role = ‘Lecturer’;

A list of Logins details with the role of ‘Lecturer’ will be produced. Looking up data on a database can also be relatively fast compared to the current systems being used.

MySQL, a relational DBMS, will be used. Relating data from two different tables at any time using joins is useful for the system. So for example, if a student wanted to view their details on our system, a query linking the Logins table and Students table using the shared UniID field would be used. By normalising the tables and accessing them via relationships, a table will only be accessed where necessary, and so data is far less likely to be altered unintentionally. This is extremely important in maintaining data integrity.

Databases have structure to them, which means instead of having to reinvent the wheel in php or python to deal with the data which needs to be stored, languages such as php and python can directly interact with the MySQL database which has already structured the data. This imposed structure also ensures that data is recorded exactly as intended, so data integrity is maintained.

Databases are also concurrent, meaning that when a group wants to check their mark, multiple members can do so at the same time without corrupting the data. This ensures that the data is the same as it was when it was originally recorded by a moderator. Again, data integrity is maintained.

Finally, what makes a database essential, is that they are fault-tolerant and can handle very large data sets. Considering that the system will be handling the data of hundreds of students and staff members, it is important that a database is used so that the system doesn’t grind to a crawl when a certain number of records is reached. As mentioned above, the system is going to be handling a lot of data and queries at times, and so the need for a robust system. So MySQL not completely breaking down in the event of an error makes it suitable for the system.

An issue faced when managing the GitHub was getting all group member usernames on GitHub so they could be added as collaborators. Initially, getting people to send their names in the first place was an issue, which was resolved by reminding people via the WhatsApp group chat, setting up a Trello card and finally reminding people in person at the next supervisor meeting. In addition to this, some team members reported that they could not see the invite to the repository. This was resolved in two stages. Stage one was reminding people to check their email inboxes which were linked to their account, and stage two was cancelling the invites and resending them, as either the invite was lost in their inbox or GitHub failed to send them an email in the first place.

When I was familiarising myself with the two systems we currently have in order to start building the database, I came across a bug within the grouping tool. The export CSV function would not save the student id field, as the private variable used within the student class mismatched the variable name used elsewhere in the program, so when it came to writing the value to file, it would always be null. I fixed this by making sure that the variable name was consistent throughout the program, checking the export CVS function then worked correctly, then pushed the fix to the correct GitHub repository.

Another issue I have experienced early on in the project, was that sometimes I would not have something to do outside of managing the GitHub, database management or report writing for example. I messaged our project manager to inform them of my inactivity due to this, who suggested I join one of the pre-existing teams, which resolved the issue.

During our next meeting I was informed that there was indeed issues with the grouping tool I could’ve been dealing with earlier. This has been remedied for the most part, but I do think in the future I need to make more of an effort to approach both teams to understand their current situation and what they need help with to avoid inactivity in the future.