Individual Report

Liam Wardle

Name	Weighting
Durgesh Pareva	33.3%
Liam Wardle	33.3%
Jack Williams	33.3%

Overall, I found the coursework to be an interesting example of implementing a database into an existing product. Having a large portion of the Java and HTML already written allowed us to focus on implementing the SQL side of the project, but it also meant we had to work within the restrictions of the API. These limitations are what I found most interesting, as it felt like how working in a real workplace would be, where part of the product would be created by a different team and the database would have to be implemented around it.

One of the biggest lessons I have learnt from this project is to fully study the API and existing code before starting. There were a few occasions where we had to alter our schema, as there were methods which needed to be implemented, but could not be using the current schema.

For example, we spent some time trying to find a way around the fact that a 32-bit int could not store a unix-epoch timestamp in milliseconds. Our initial solution involved storing it as a string and converting it later. Had we fully read the source code provided, we may have seen that the time stored was being multiplied by 1000 by the application. This meant that we could store the time in seconds, rather than milliseconds, which would fit in our 32-bit limit.

Other things became easier as I became more familiar with JDBC. For example, learning I could insert a null value into a primary key and have it automatically generate made inserting new records much simpler.

The methods I found most challenging were those which involved multiple queries, as it could be difficult to keep track of where to catch errors. For example, creating the main page which lists the forums and their most recent topics proved challenging, as it involved querying the forums and then all topics inside the forum. What's more, we also had to handle forums with no topics, which added to the challenge.

If we were to do this again, I might have spent more time on these methods and attempted to perform them in a single query.

I found JDBC quite straightforward to use and was able to work out most things from looking up the documentation online. If I was to design my own API, I would use JDBC for its simplicity and its place as the standard implementation.

A lot of methods in the API took non-key values such as “username”, and these often required looking up the primary key associated with them. If I was to write my own API, I might have considered changing this so that primary keys were passed to the API instead. However, this would mean that the HTML side would need to get key instead, so this just passes the responsibility to someone else. As this might not be someone familiar with the database, it might be better to not change this part of the API. It does mean slightly more work needs to happen behind the API, but at least this keeps all of the database related tasks in one place.

One thing I definitely would change is the database language. SQLite is not really suitable for this task, as there is no concurrency support or role-based access control.