University of Toronto

Phase 3

CSC301

18/11/2014

Group Members:

Amrutha Krishnan
Binuri Walpitagamage
Carl Gledhill
Michael Li
Sang-Ah Han
Tommy Li

Why are you changing your design?

As a group, we have decided to keep our current design as originally laid out in Phase I. This design was methodically planned after consultation with our entire group over the course of many group meetings. While reflecting upon the implementation of Phase II during the last meeting of our second sprint, our group agreed that the design had worked well. Furthermore, we believe our design is perfectly structured to meet the goals and objectives of our future sprint implementations. One additional feature we have decided to include in our final Phase which originally we had excluded is the notification feature. The group has decided to include the notification feature in the final project after completing more work than anticipated in Phase III.

Briefly highlight the problems you have faced and how as a team you managed to solve them.

Loss of a group member

One of the largest hurdles our team had to collectively overcome was the loss of our seventh group member Matthew Shwed. When Matthew initially joined our team he was very enthusiastic and appeared eager to work on the project. During our first planning meetings the group debated which frameworks and languages we should code the project in. While many members of the group were leaning towards using PHP for handling the server-side work, Matthew insisted that we should use Python coupled with Django framework. While some members in the group displayed hesitation about using a framework that only Matthew had experience in, Matthew assured us he had a plethora of experience using Django it was the best direction to take. Matthew was even kind enough to allow us use of his Ubuntu server. As the weeks progressed Matthew seemed to withdraw from the group; he stopped answering messages, would not answer phone calls, and did not push any code for Phase II. After Matthew did not write the midterm it became increasingly obvious he had dropped the course and yet did not have the courtesy to inform his group. As the deadline approached our group realized we would have to carry on without Matthew by dividing up his tasks into smaller tasks and reassigning them. Our team doubled down our efforts and were able to finish our Phase II by the due date. While, the unexplained and unexpected loss of Matthew was a significant challenge our team faced, it showcased the drive and determination of our group members in the face of adversity.

Lack of experience

As mentioned above, none of the active members of our team had any experience with using the Django framework. As a group we were able to overcome our lack of experience by completing Django tutorials and reviewing best Django coding practises. Our group also utilized Facebook chat to help one another fill in the shortcomings in each other's Django knowledge as specific problems cropped up.

Dividing up work & combining the project

Another large problem out group faced was the division of labour within our group and how the project should be split up. The team decided an effective strategy would be to split the team up into two smaller teams with one team focusing on front-end development and the other team focusing on back-end development. While we found this to be an effective division of labour, we ran into some issues when trying to link the front end to the back end. This crucial problem of synthesising our collective efforts was overcome thanks to the remarkable work of our front-end team. The front end team ensured each of their templates matched a back-end view, when the front-end team found errors or inconsistencies they asked for clarification from the back-end team and registered a bug through GitHub's issue tracking system.

Highlight Design Patterns

Model-Template-View (MTV)

The Django framework implements a MTV design pattern, this design pattern shares many similarities with the more commonly used MVC design pattern. Django's framework itself acts as the "controller" itself by sending requests to the appropriate view, according to the Django URL configuration. Django's "views" act as an interface between the model functions and the template. The traditional MVC "views" are similar to MTV "templates", both describe how data should be presented to the end user on the front-end.

Database access object (DAO)

The Django framework provides a QuerySet object for each model. This object enabled us to obtain relevant data from our database system (MySQL). One of the key advantages of using the QuerySet interface is its ability to allow us to construct the system while abstracting the underlying details of the database. This allows our code to

be flexible, readable and independent from the intricate structural details of the underlying database.

General Design Principles

As a group we strived to implement our project adhering to best S.O.L.I.D. Object Oriented coding practises. Our group has a strong dedication to building code which is not only functional, but crucially, also well laid out. By ensuring each class and method was well defined in both scope and functionality, our group was able to create easily read code, which is critical when many members are editing the same functions. Furthermore, since the code was designed with the “Interface Segregation Principle” in mind, the code will be easily extendable when our group begins Phase IV.