Codeline

Code sharing platform

# Analysis

* Specifically for school, dept
  + Growing CS dept, introducing CS at GCSE and A level
* Problem: accessing, keeping track of, storing (centralized), finding, checking and marking students’ code
* Process/stages of development
  + Iterative process with feedback from MWD, other students
  + User testing, amendments
  + Diagram?
  + Deployment
* What I did to find out more about problem
  + Interviews, talking to students and teachers
* Current system – putting code in collaborative area (Y drive)
  + Time consuming
* List of client requirements plus possible limitations (e.g. time, hard work for little gain)
* List of objectives that are realistic, achievable and measurable

## Background

Dubai College’s Computer Science Department recently started offering Computer Science at GCSE and A-Level. With the expanding department, and a lot of growing interest in the subject, the teachers of the subject are increasingly pressured by various different responsibilities, including keeping lessons engaging and students interested; developing projects and problem sets to teach in ways other than textbook and PowerPoint; and consistently marking students’ work to give feedback to students. In the experience of Mr. Mark Wood, the head of the CS department at DC, the third point is often defaulted on, as a lot of energy goes into the other parts of teaching and not much is left for marking. As a result, students are often not getting the tangible feedback they need to improve their skills as computer scientists, and Mr. Wood and his colleagues often lose track of their students’ progress. Of course, tests can be used to gauge progress, but specifically for CS, it would be beneficial for the teachers to see the code that students are writing outside of exam conditions—programming is, after all, a large part of the subject.

I had a talk with Mr. Wood that illuminated many of the problems the department is currently facing in regards to marking. He told me that he genuinely wants to mark students’ code, as he often gives programming tasks to the class for which he would like to observe each individual students’ approach. Looking through the code of each student would give a better understanding of each student’s progress, and insight into the effectiveness of certain projects or teaching styles for the cohort as a whole. Unfortunately, he explained, the marking often doesn’t happen as it is too time consuming. There are a few approaches he has tried, but each is suboptimal and fails in certain ways to match his workflow.

The first option is to get students to print out all of their code on paper and hand it in, much like perhaps a History essay would be presented. This works in getting all the code in one place, however is not optimal for a CS class which is trying to move towards a fully digital atmosphere (and, of course, minimize deforestation). Mr. Wood also expressed that he would much prefer digital submission as it is easy for students to lose track of old work in physical form. A digital submission method would allow both students and teachers to find work from any point in the year at any time, and access it both at home and school without needing to carry heavy physical books or folders.

Another option that was explored was having students put their work on their school network drives. This does allow students to keep all their work in one place, and allows teachers to access work for marking without need for physical paper, however is still significantly inconvenient as the teacher has to individually access each student area and locate the relevant file manually. This can be especially taxing when some students miss deadlines and so teachers spend time looking for work that isn’t there. The difficulty of accessing work that is not in a centralized location is especially visible with this approach.

The obvious solution to this is to ask students to submit their work to a shared area, perhaps on the school network or a cloud file sharing service such as Google Drive. Again, this allows students to keep

# Documented design

* Explanation of Django, MVC system
  + URLs and views
  + Templates
* More detail about database
  + What a post looks like
  + Diagram showing post’s anatomy
  + How posts are read and rendered
* Detail about libraries: Ace, highlight, handlebars
* Screenshots/diagrams of basic UI
* Exploration of logic behind some views

# Technical solution

# Testing

# Evaluation