COMP204/242-12B
Assignment 2

Mathew Andela
Joshua Scarsbrook
Janik Singh
Nigel Thomas
Duncan Willcock

The purpose of this development effort is to produce software intended for use in education allowing a specialised mechanism for students to submit work and teachers to view, mark and organise their students' work.
While there are plenty of existing examples of electronic submission mechanisms, these often lack specialisation, are only a small part of a much bigger system with a broader purpose and still unnecessarily require traditional 'pen and paper' methods during the processes of assessment. With increasing trends in all areas of society to favour efficiency and sustainability, paperless tools and methods are becoming more and more accepted. Schools and universities are no exception and it is not unreasonable to assume that many would readily adopt such tools were it adequate to the purpose. Furthermore, such tools would be invaluable in increasing the capacity of distance learning institutions and correspondence schools.
To cater for these requirements, this software provides a tool specialised to perform the tasks related to assessment all accessible using a web browser for ease of use and high accessibility in all important respects. Students will be able to upload files and submit work from any computer and their educators will be able to view these submissions and mark them from anywhere. Furthermore, administrative functions will be catered for with the ability to view marks received and collate reports and similar such documentation.
The important considerations that underlay all decisions made during this development are simple:

- The product must conform to all requirements laid down during planning
- The product must be accessible and usable in its interface
- The product must have been rigorously developed and tested to avoid issues and irregularities
- The product must be secure so users are comfortable using it

While these may seem obvious, it is important to state them explicitly here as every decision made during the development process must move towards achieving these goals. A decision that compromises any of these is not the correct decision.

Timeline

The projected timeline for this project over approximately 2 years, which is predominantly split into two week periods representing a single development iteration. At the very beginning of the project is an initial planning process which has several specific purposes:

- Develop the initial product idea
- Break this informal, generalised planning into a list well-defined component functions the product requires
- Set the timeline for the project
- Discuss and refine a development methodology suitable to be followed throughout all stages of development

Most of the rest of the project consists of iterations of development: two week periods with clearly defined goals and a clear structure. These are split more broadly into groups that focus on related but different functions within the software.

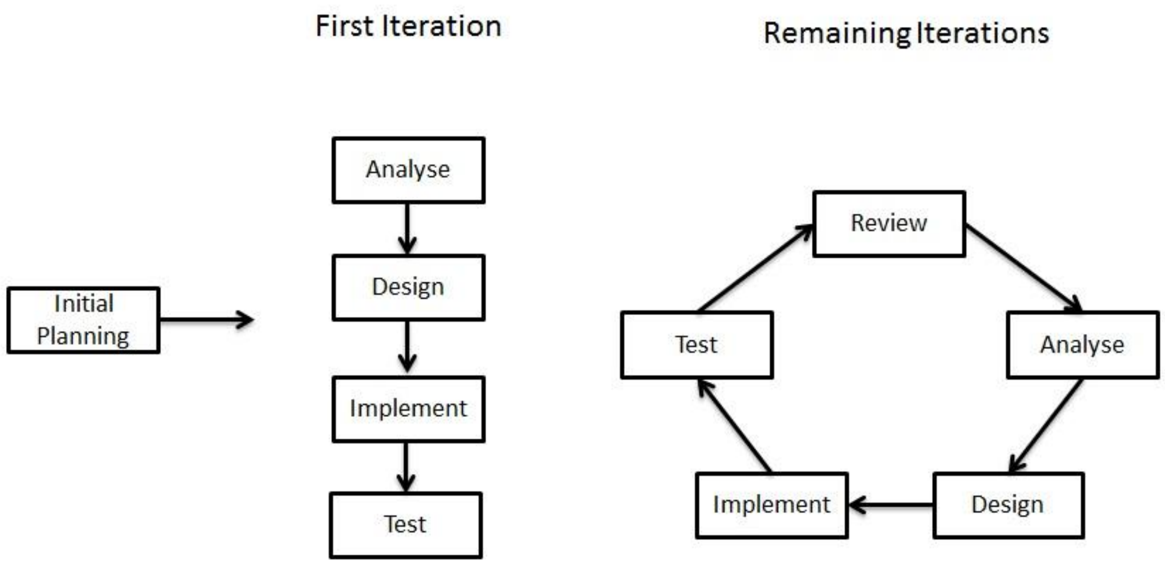
Interspersed, and as a complement to, the regular development iterations are several periods allowing for evaluation of progress to date. One major intention of these one week long periods is to gather information about how members of the intended user base use the product in its current form. This is primarily to enable the design of a highly usable interface, adapting and reworking constantly to better adhere to the needs and capabilities of intended users rather than making erroneous assumptions throughout the majority of the development process and attempting to later improve the user interface. This would add inexcusable time pressures at the end stages of the process.

Additionally, frequent communication with real potential users may reveal small but relevant, otherwise imperceptible errors or omissions from planning, analysis or design that could and often should be added into the appropriate phase of the development process.

The actual schedule for the project is presented in Timeline.pdf.

Methodology

The methodology chosen to be used primarily utilises many consecutive, relatively short iterations of two weeks. Each of these periods has a set goal or milestone that it is hoped will be reached by the end of the iteration. Effective communication is achieved in a formal setting with daily full team meetings set into the structure of each iteration.
Each iteration period will start with a meeting (1-2 hours) to review progress or problems from the previous iteration, review the intentions set for the current iteration, set the goals for the upcoming two weeks and allocate tasks to members of the team. Ideally these goals will match exactly those set during initial planning but it will likely become unavoidable that some features are pushed back. It is important to mention here that only functions that are fully complete are acceptable for release at the end of iteration.
In order to track overall progress and allow all individuals to maintain a awareness of the context of their own work, each day will be started with a short (15-20 minute) meeting. This time is extended slightly on the day in the middle of the 2 weeks in order to make some pre-emptive assessments as to what functions will be complete, what may be delayed and the magnitude of any issues.
The iteration ends with a milestone release consisting of adding everything totally completed during the iteration to the current release.
During the two weeks of an iteration individuals or pairs (depending both on the magnitude of the tasks and the particular individuals involved) will work on allocated functions. In general the process used proceeds through Analysis, Design, Implementation and Testing stages in that order. This will not necessarily take place all in one iteration with some functions simply too big to fit in a two week period. Furthermore, this is not a ‘waterfall’ methodology; stages of development will need to be returned to due to new information or issues discovered during other stages of development.



Testing

Testing is vital stage of this process and it needs to be conducted robustly to ensure control of quality rather than enacting a pointless ‘testing’ process that does not in fact test the product against the full set of defined requirements. For effective testing, throughout development it should be the team’s policy that no-one tests their (or defines tests for) own code. This is to avoid the conflict between an individual’s emotional investment in their own work and achieving the goal of quality software. As iterations pass, more and more tests will be added through the growing test suite to prevent the solution regressing to display previous errors.

Decision Making Process

Decisions need to be shared but we also need to allocate our time efficiently. Ensuring every team member fully understands all facets of the project and can therefore share fully in all decisions is very time consuming. Therefore we will not require everyone to agree to all decisions but in order to avoid poor decisions making there needs to be some level sharing depending on the magnitude of the problem at hand. Therefore, the rule of thumb during this project will be to always seek input from one other team member able to understand what has been done and what has been proposed. This ensures everything is seen by another individual who may be able to use a different perspective to remove flaws. The result of such consultation may be the decision that a problem is big enough to be handled in conjunction with others or maybe even the whole group.

Documentation

Completing work is of the priority, but some record must be made especially looking ahead to later iterations. At the core of documentation procedures is the log for each iteration. A brief but comprehensive record of the artefacts produced, changes made and comments on consequences or implications on development going forward. Other important documentation is code comments which need to complete but also succinct.

Tools and Technologies

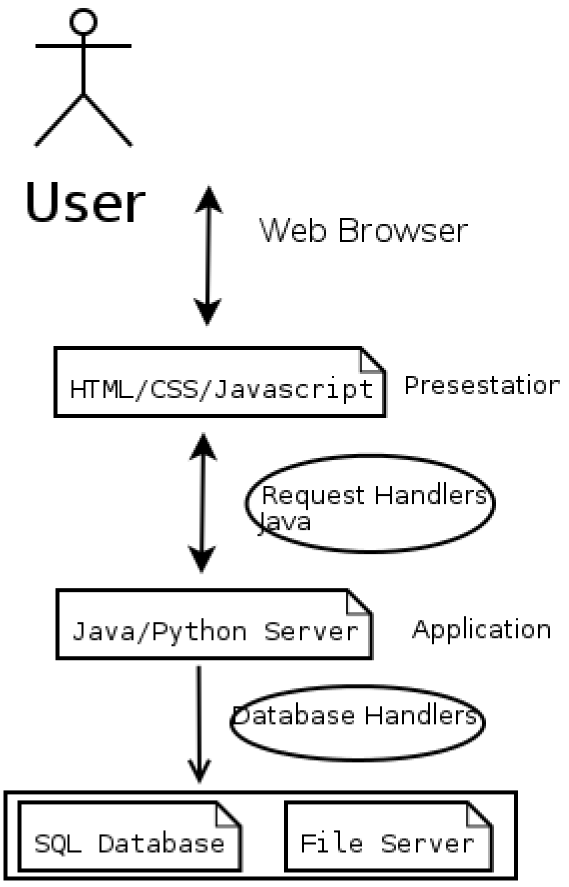
Different aspects of the application will require using a diverse set of tools, languages and protocols:

- Java for programming core application layer and communication components
- Python 2.7 for extensible server-side scripting
- MySQL – Database Management System for the underlying database system
- Linux – operating system for database and file servers
- HTML 5 and CSS for web interface presentation
- Javascript for client-side functionality
- HTTPS for client-server communication
- FTP for application communication with file storage

During development several tools will be used:

- JUnit package to test Java code
- Test.py library to test python scripts
- Eclipse as an IDE to uniformly manage source code at a personal level
- Git (Github.org) for source control and version management
- A wiki (Github.org) for documentation

Architecture
The proposed application will be designed using a three-tier architecture:



- The *Presentation* tier is the interface presented to the user, here through a web browser and web pages generated by the server fitted to predefined templates.
- The *Application* (or *Logic*) tier provides the bulk of the processing receiving requests from users, processing these, finding or changing the necessary data and generating appropriate responses to return to the client.
- The *Data* tier is the simplest, it is where persistent data used by the application can be stored. This is chiefly achieved with a relational database and a file server as this application requires the secure storage of large quantities of files.

Risk Analysis

This project is planned to take a very time in terms of software development a fact that greatly compounds all sources of risk. Most if not all of these risks are inseparable from the processes of the business of software development but with early identification and comprehensive planning, these can be minimised.

- Attrition of team members. There are many possible eventualities over 2 years that may prevent a member of the development team from working on the project permanently or for an extended period. Significant loss of productivity or the skills of a particular individual will set the project back.
- Running out of time. If the initial planning has omissions or simply underestimated the amount of time different things would take delays could cause the project's overall failure.
- Taking shortcuts and falling to the temptation of using bad practices risk the overall quality of the end product. Following the defined design and development principles and methodology is the best way to avoid this. Shortcuts may save work in the short term but create more in the long term.

Other areas of risk are certainly significant but less necessary or consideration from a development point of view are financial and product/marketing development concerns. While these are very real, it is not the job of the software developer. The job of the software developer is responsible for creating a usable piece of quality software but marketing a product is not.
Wasteful use of resources could lead to the continuation of the project becoming impossible. This presents a need for efficient allocation of resources and diligent management of finances.

Requirements:
Functions identified during initial planning:

Login
Upload File
Download previously uploaded file
Allow re-submitting (teacher option)
Create an item of assessment
Create a mark sheet for that
Add\Remove teacher to class
Add\Remove student to class
Add\Remove class
Display student deadlines
Provide receipt on upload
Logout
View submissions to mark
make annotations to file
add marks to submissions
Class mark Report
Allow or disallow students to see reports
Individual student mark reports
Email notifications of deadlines and marks
Reveal marks
Change marks
Teach can specify amount and size of files uploaded (E.G. 1 .java file)
Close submissions at deadline
Change Deadline
Plagiarism checking (Comparing differences of files, and marking similar files)
Mark distributions report
Add Report to class
Set visibility of a report
Mark distribution graph
Add a distributions to a class
Class comments
Add marks for not tracked stuff like tests
Add\Remove Tests\Exams as assessment items
Send Message to Student\Teacher
Enable or disable messaging system per class
Read Received Message
Set an email address to forward messages to
Enable\Disable Email notification for an account
Search a phrase or keyword from file
Create\delete spectator account
Associate/dissociate a student with a spectator account
Display associated student data to spectator