Mark Tracker

# Business Context

Marking assessment tasks are a chore, especially when it comes to book keeping information. There are 3 primary pieces of information that are essential to the marking process:

* Students - those who submitted
* Marks - the raw marks obtained by each student
* Feedback - suggestions for how the student could improve, or where they lost marks

Complicated assignments make this information hard to retain in a systematic way, which as a result makes it difficult to book keep and relay the results back to the students in an efficient manner.

"Mark Tracker" is a GUI desktop application that shows the **structure of an assessment tasks**, in terms of its components. These components are defined by the assessment task type - for example, if the assessment is a quiz, then a component could be a question, or a sub-question.

With each assessment task, and each associated components, the application is also able to show **marks achieved by students**. For the assessment task, the application can display some statistics related to this matter: average mark, best done component, and mark distribution.

The application also allows users to preserve **feedback**. This feedback can be related to the assessment, components, or the student for that assessment. For the latter, the application allows users to preserve feedback for each student for EACH component in the assessment task.

This application was made to ease the feedback distribution process, so that the markers don't need to scramble through an assessment or paperwork to determine specific feedback or marks for a student. By preserving this information on the fly, the application gives a systematic way of achieving this task. The end users are ultimately teachers who are marking assessments, who wish to preserve, retrieve, and examine information in a systematic way.

# Requirements

Core and optional? Future? VERY FAR into the future?

### Core requirements

* Creating data  
  Users are able to create assessments, components, groups, students, and SMI.
* Viewing data  
  For each assessment, users can view its details and the associated components.  
  For each component, users can view its details and the associated groups.  
  For each group, users can view the associated students.  
  For each student, users can view their associated SMI's.  
  Users can view an individual student's SMI for particular assessments and components.
* Users can change the view of the assignments. For example, sorting them in order of date due, or weighting. They can also select what entities to view regarding the assessment task. For example, they can "toggle" to view components, groups, and students. This way, they are able to customise the depth of information they'd like to see. (See GUI for more info).
* Modifying data  
  Users can modify details (through text editing fields):
  + Assignments: Due date
  + Component: Mark available
  + Group
  + Student
  + SMI
* Removing data  
  Users can remove assessments, components, groups, or students.

### Additional requirements

* Users can view a graphical distribution and statistics of marks for a particular assessment, with regard to a component, group, or students.
* Users are able to attach assessment task specification documents (eg uploading PDF)
* Users can create a list of students by importing all their IDs
* Users can export assessment results in certain formats (eg csv)

### Future

* Make the application into a web system, with two parts: a central repository of assessments, components etc that contains all the information, and a client-viewing side (the application developed will be the prototype for this part).
  + This will also require a 'log-in' mechanism, so that each teacher has access to their own classes etc.
  + This also supports "sharing" of information (ie permissions) of which assignments are viewable / changeable by certain teachers.

### VERY FAR into the future

* Develop a compatible "collection" system, where students can collect their given marks. The end users will be the students themselves.

# GUI Design

Screenshots, flow between each screen

# System Design

High level system design (eg database or not?). UML Design: classes for each terminology, Technologies that will be used for implementation, Rules

### Rules and Assumptions

* 1-1 relationship btwn Assessment and Group
* Student ID's must be unique within an assessment. Students NEED to be created for every assessment (ie system shouldn't store separate student information - rather, they store separate assessment information, which contains students).
* When a student is created inside an assessment, they will appear in every component
* Group names must be unique within an assessment
* When a group is created, they will appear throughout the whole assessment

### Function rules

* Creating an assessment:
* Deleting a student: Student will be associated with one assessment. Thus removing the student will remove their presence in all associated components,
* Exit button: Will prompt to "Save" if there is unsaved modifications

# Testing

Methods of testing

# Terminologies

Terminologies

* **Assessment**: An assessment task. It could be a quiz, or an assignment, or something else. It should be able to be split into components. Contains important information such as due date, weighting,
* **Component**: Individual parts of an assessment that are worth marks. Differ based on the assessment type in real life. For quizzes or tests, a component is a question. For assignments, a component is a task / deliverable.
* **Subcomponent**: A sub-part of a component. For a question in a quiz, this could be "part a".
* **Group**: A collection of students. In real life, this could be a class or a team. Associated with an assessment.
* **Student**: An entity who is being marked, and is associated with a group. They may also have various SMI's. Information held: name, ID
* **SMI**: Student Mark Information. Contains a student's mark and feedback.

# Questions

Feature requests:

* Feedback not ONLY for the student, but for the assessment? For the component? For the group?