Chapter 1

User Manual

1.1 Basic Concepts

1.1.1 Quiz Type

This software supports two types of objective question quizzes:

1. **Simple quiz:** In this quiz, the students are supposed to answer all questions. All students answer the same question paper.
2. **Jumbled quiz:** In this form of quiz, each student gets a different question paper to answer, which contains questions sampled out of a larger item bank and jumbled.

1.1.2 Question Types

There are two types of questions that can be included in the question papers: multiple choice questions (MCQ) and match the following (MTF). An MCQ is a question with two or more possible options out of which one or more may be correct choices. For example:

> Name two dynamically typed programming languages:
>
> 1. Java
> 2. **Python**
> 3. Haskell
> 4. **Ruby**

In the above, options 2 and 4 are correct answers.
The second question type is match the following.

> Match the following programming languages on the left column with properties on the right:

1. Python	A. Static typing
2. Java	B. Dynamic typing
3. C++	C. Untyped
4. OCaml	D. Implicit typing
	E. Explicit typing
	F. Functional
	G. Object oriented

In the above example, the matches are as follows

1. Python matches with B., D., F. and G.
2. Java match with A., E. and G.
3. C++ match with A., E. and G.
4. OCaml match with A., D., F., and G.

A few points to note here:

- The mapping can be many to many, even allowing some options on the either side to have no matches at all (e.g. C.).
- There is no restriction on the number of options on either columns. For example, in the above example, we have 4 options on the LHS and 7 on the RHS.

1.2 Simple Quiz

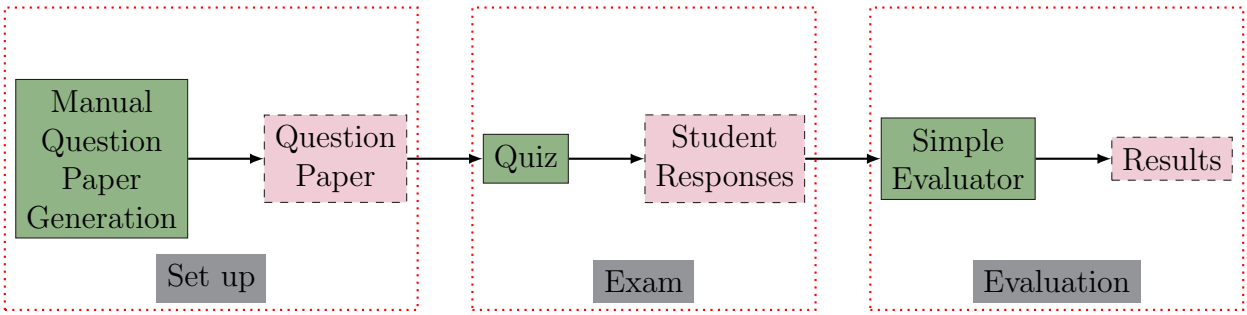

Figure 1.1: Simple Quiz Workflow

The question paper creation for a simple quiz is manual. All students solve an identical question paper. The evaluation step directly runs on the student responses using the `SimpleEvaluator` module.

1.2.1 Setup

Suppose you intend to conduct a simple quiz. As mentioned above, in a simple quiz, all students solve the same question paper. Such a quiz is ideal when there is small class and enough assurance that cheating is not a possibility. Of course, all questions are assumed to be either multiple choice questions (MCQ) or match the following questions (MTF).

The assessment project can be setup conveniently using the setup utility. The setup utility automatically generates the skeletal infrastructure for conducting such a quiz.

1.3 Jumbled Quiz

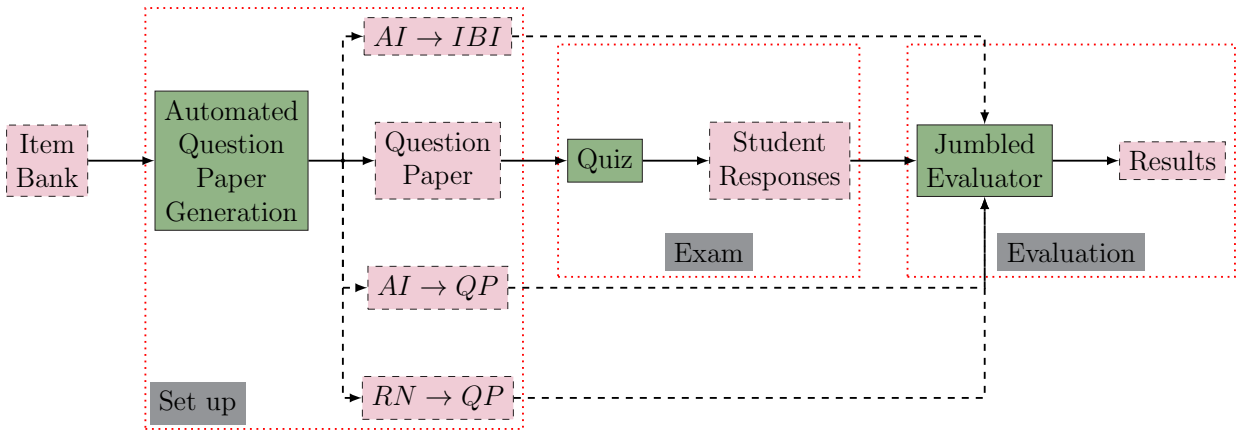

Figure 1.2: Jumbled Quiz Workflow

The question paper creation step uses the `genAIs` (generate assessment items) and genQPs (generate question papers) modules. Once the question papers are generated, quiz is conducted. Finally, the automated evaluation process takes place using the `JumbledEvaluator` module.

Chapter 2

Design

2.1 Question Paper Codes

2.1.1 Question Paper Generation

To discourage cheating in the class, we generate a set of question papers by randomly selecting n questions out of an item-bank of N questions. A set K of distinct *assessment instruments* are generated, numbered 0, 1, ..., $|K-1|$. We call them *assessment instruments*.

The *question paper generator* module G generates a set C of codes each of which can be mapped to any one of the assessment instruments of K. Each of the code c in C is finally mapped to one distinct question paper with c printed on it. This way, the students will not be able to identify which assessment instrument $k \in K$ their copy of the question paper belongs to. Each question paper will have an empty table called the *response table* on page one which will be used by the student to fill in his responses.

G also generates a map from assessment instrument to question order. This tells us the original question number of each item in the give assessment instrument. For example:

Figure 2.1 shows a possible mapping from assessment instruments to item bank items. The table can be interpreted as follows: There are 10 assessment instruments numbered 0 through 9. For each assessment instrument $AI \in K$ (here $|K| = 10$), there is a row in the table. Each cell in that row has the item bank item number for that item. For instance, for $AI = K[0]$, $AI[0] = 3$, $AI[1] = 5$ and so on.

This module will generate a map – called $QP \mapsto AI$ between *question paper code* to *assessment instrument*. For example:

$QP \mapsto AI = [0, 1, 2, ..., 9, 0, 1, 2, ...]$ could be one such mapping. It says that $QP \mapsto AI[0] = 0$ (i.e. the question paper with code 0 maps to assessment instrument number 0). Similarly, $QP \mapsto AI[11] = 1$ (i.e. the question paper with code 11 maps to assessment

0	3	5	1	4	9	12	15	2	10	2
1	4	1	2	11	6	7	5	14	8	12
	...									
9	5	12	2	1	6	7	3	11	8	10

Figure 2.1: Assessment Instrument Item to Item Bank Item map $AII \mapsto IBI$

instrument number 1) and so on.

2.1.2 TA's Job

The TA will note down following:

1. Question paper code for each roll number creating a *roll number* to *question paper code* map $RN \mapsto QP$.

2. transfer the responses into a CSV file corresponding to each student exactly as in the response table.

2.1.3 Automated Evaluation

The *response rearranger* refers to the $RN \mapsto QP$ and $QP \mapsto AI$ map to extract the assessment instrument for each roll number. Using this, the evaluator rearranges the responses in the order as per the item bank to create a rearranged response for the roll number n, $R'(n)$. This is given to the evaluator for final automated evaluation.