PES University, Bengaluru

(Established under Karnataka Act No. 16 of 2013)

100 Feet Ring Road, BSK 3rd Stage, Bengaluru-560085

DESIGN PATTERNS (UE14CS413) PROJECT REPORT
ON

" QUESTION PAPER SETTER"

Submitted in the partial fulfillment of the requirements for VII Semester

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING

Submitted By:

ARVIND BASKAR(01FB14ECS047)
RAHUL RADHAKRISHNAN (01FB14ECS179)
VII Semester, B.Tech, CSE

Under the guidance of:

Prof. N S KUMAR
Dept. of CSE
PES University, Bengaluru

During the Academic Year
2017-2018

Description

The project presents a software that helps professors select questions from a very large question bank. Questions can be loaded from any directory in the system. The user can interact with the system via a simple, menu-driven CLI.
The user can chain queries to select the questions based on:

1. Marks per question
2. Difficulty of each question (on a certain scale)
3. Keywords contained in the question

The final collection of questions can then be written into a file in increasing order of marks per question for persistence.

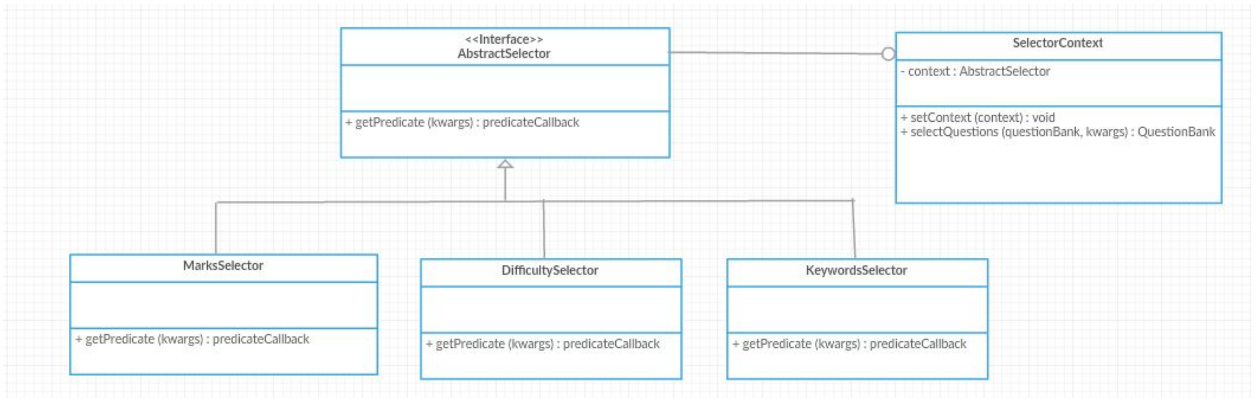
Design Patterns Used

Strategy Pattern

The strategy pattern is used to implement filtering questions out of the question bank based on three different criteria. On future development, additions to the ways of filtering the questions can be handled effectively. There are three different strategies employed to select questions as mentioned in the previous section.

Iterator Pattern

The iterator pattern is used to easily walk through the collection of questions. This makes it convenient to use ***filters***, ***maps*** and ***for*** loops on the collection..



<<Interface>>
AbstractSelector
+ getPredicate (kwargs) : predicateCallback
SelectorContext
- context : AbstractSelector
+ setContext (context) : void
+ selectQuestions (questionBank, kwargs) : QuestionBank
MarksSelector
+ getPredicate (kwargs) : predicateCallback
DifficultySelector
+ getPredicate (kwargs) : predicateCallback
KeywordsSelector
+ getPredicate (kwargs) : predicateCallback

Fig.: UML of the Strategy Pattern in use

Implementation

Language: Python 3.x

The user is first asked to input a directory path to a bank of questions. The system then walks through the entire subtree rooted at that directory, looking for files with questions in them. It then loads each question into a ***Question*** object and puts all of them in a ***QuestionBank*** object which serves a collection. The iterator pattern is used on this object.

Then, based on user's choice, a sequence of filtering is performed. For this, the client script talks to the ***SelectorContext*** object which stores the context of the strategy implemented. The different strategies are used to return different unary predicates back to the context class. And that predicate will be used in a ***filter*** to walk through the QuestionBank and pick relevant questions. These selected questions are put into a new QuestionBank and returned back to the client.

The strategies of ***MarksSelector*** and ***DifficultySelector*** are straightforward. They filter based on a numeric range.

The ***KeywordsSelector*** uses a few basic NLP principles. Every question is tokenized and all stop words are removed from the question following which all the tokens are stemmed and a set of tokens is stored in each Question object as potential keywords. Input keywords are then matched with this set of potential keywords and based on a threshold ratio and number of input keywords

matched, we decide whether to select a question or not. Python’s ***nltk*** library helps achieve this functionality.
Finally, the QuestionBank will be serialized into a text representation and written into a file for persistence.

Future Developments

1. Integrate with an online test platform to make the whole system of examinations automated.
2. Update the values of the difficulties of the questions based on statistics of exam results.
3. Integrate with a web crawler constantly on the lookout for interesting new questions on relevant topics.
4. Implement various analytics algorithms like for topic modeling so that selecting questions on related topics becomes easier.