MacQTEX: Online self-marking Quizzes, using pdfTEX and exerquiz

Ross Moore
Mathematics Department, Macquarie University, Sydney
ross@maths.mq.edu.au
http://www.maths.mq.edu.au/~ross/

Frances Griffin
Mathematics Department, Macquarie University, Sydney
fgriffin@maths.mq.edu.au
http://www.maths.mq.edu.au/~fgriffin/

Abstract

The MacQTEX quiz system uses JavaScript [1, 9] embedded within PDF format [4] documents to allow students to do multiple-choice style quizzes. The internet may be used to supply the quiz document, and to record results. But even when not connected, there is immediate feedback as to how many questions were answered correctly and what are the correct answers, as well as providing worked solutions indicating how the correct answers could be deduced.

The highest quality of typesetting is employed in the quizzes by using the TEX typesetting software [7], via the pdfTEX variant [6], to control the generation of the PDF documents [4]. Other software, such as Perl [12] and *Mathematica* [13], can be used to control the production of unique instances of a particular quiz so that each student gets slightly different questions to answer.

See also the presentation slides (PDF).

Preprint

PDF Quizzes

At Macquarie University the Mathematics Department has been developing[1] a web-based system for producing quizzes which allow students to test their knowledge of mathematical ideas commonly used in courses that we teach. Currently these quizzes are used mainly at the most elementary level, for revision of the basic skills which the students should have acquired from courses at high school.

The current version of this quiz facility provides students with a multiple-choice answer quiz, of typically 10–12 questions, as a PDF document [4] downloaded from a web-site (figure 1). This document is an interactive form, controlled using embedded JavaScript [1, 9], which allows a student to read and work with the document, using the Acrobat Reader plug-in [2] to his/her favourite web-browser (figure 2). Figures 2–7 show some views of such a quiz, as it appears to the student before, during and after attempting to answer the questions.

In this paper we will concentrate mainly on the TEXnical aspects of the MacQTEX quiz system. For other aspects of the full system, such as the rationale for using quizzes at all, and features available to an instructor when preparing a set of quizzes for use by students, figures 13–15 show presentation slides prepared[2] for talks at educational meetings.

pdfTEX, exerquiz and JavaScript

A quiz document is typeset using pdf-LATEX [6], with the exerquiz [10] macros to handle the embedded JavaScript [1, 9] actions needed to produce appropriate interactivity. In this setting, JavaScript controls

- the appearance of check-boxes, as the student selects his/her answers;

[1] This project has received funding via a 'Targeted Flagship Grant' from the Center for Flexible Learning, Macquarie University, and the Division of Information and Communication Sciences, Macquarie University as well as an equipment grant from Apple Computer, Australia Pty Ltd, via the Apple Universities Consortium.

[2] ... using the Marslides package [8].

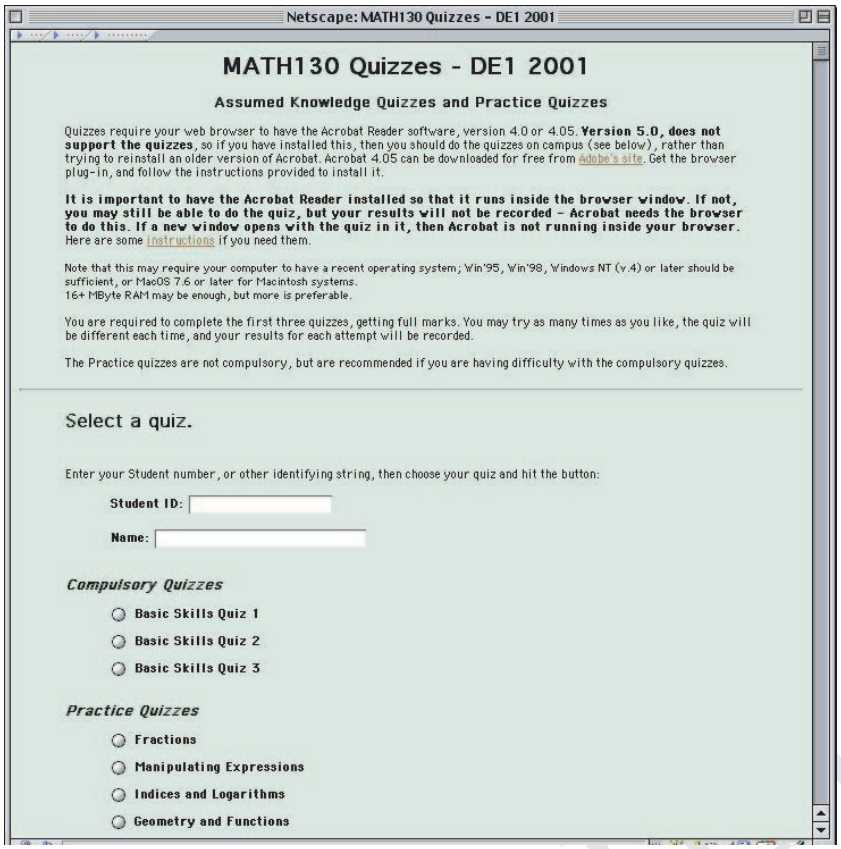


Figure 1: Quiz-site (at http://www.maths.mq.edu.au/~fgriffin/quizzes/MATH130quizzes.html) from which students can download the compulsory quiz documents. Username/password are required for recording accesses and results. Also from this site they may download practice quizzes, devoted to a particular mathematical concept. Guest access is also allowed for all quizzes.

- counting the number of correct choices selected, and displaying an appropriate message;
- showing which of the student's selections were correct, which were wrong, and which were the correct choices for each question;
- resetting the form, for further attempts at the same set of questions.

Donald Story [10] has explained some of these methods elsewhere in this volume.

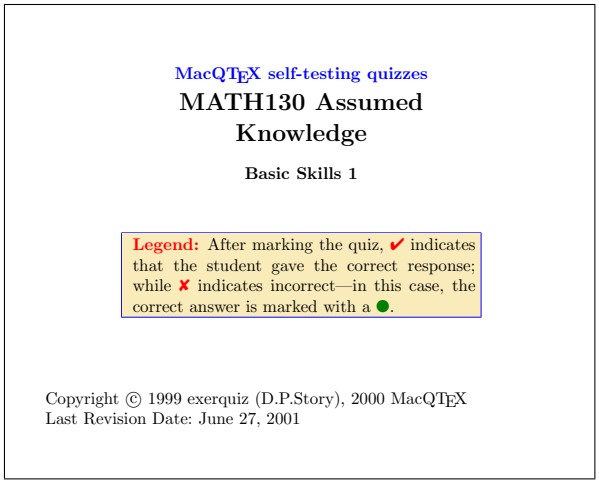
In fact we have added some extra features not found in the released versions of exerquiz. Hence the the macro file that we actually use is named exerquizX, and epdftexX has the coding specific to the pdfTEX driver. These extra features include:

- a quiz variant including both the self-marking feature *and* worked solutions;
- submission of a student's results to a server, via the Internet;
- return of a personalised message, as feed-back to acknowledge receipt of the submitted results.

With these features, a quiz document provides a nicely typeset collection of questions and solutions which can be studied by a student, even when not connected to the Internet. The quiz can be reset for repeated attempts at answering the questions. Only the first attempt can be recorded, and even then only if there is an active network connection.

Concerning the worked solutions, these must remain inaccessible until the quiz has been completed and the results submitted. To do this, the worked solutions are typeset in a separate document, then each is imported as a separate image to be the icon for a button field. This button can be shown or hidden under the control of embedded JavaScript code. As well as allowing the solutions to be viewed, a hyperlink from

Preprint


MacQTEX self-testing quizzes

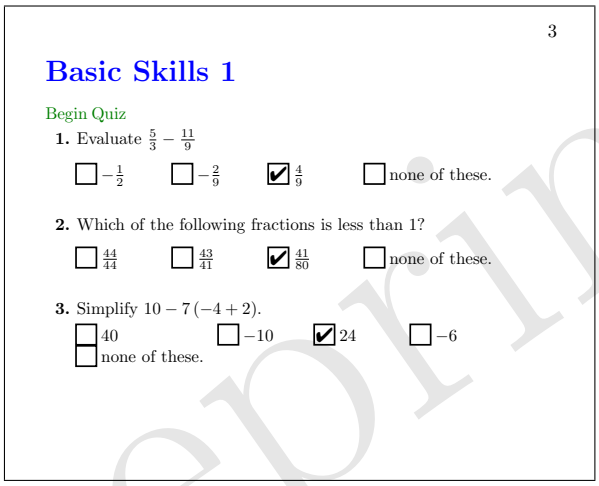
MATH130 Assumed Knowledge

Basic Skills 1

Legend: After marking the quiz, ✔ indicates that the student gave the correct response; while ✘ indicates incorrect—in this case, the correct answer is marked with a ●.

Copyright © 1999 exerquiz (D.P.Story), 2000 MacQTEX
Last Revision Date: June 27, 2001

Figure 2: An opening page to a typical quiz.


3

Basic Skills 1

Begin Quiz

1. Evaluate $\frac{5}{3} - \frac{11}{9}$

☐ $-\frac{1}{2}$ ☐ $-\frac{2}{9}$ ☑ $\frac{4}{9}$ ☐ none of these.

2. Which of the following fractions is less than 1?

☐ $\frac{44}{44}$ ☐ $\frac{43}{41}$ ☑ $\frac{41}{80}$ ☐ none of these.

3. Simplify $10 - 7\,(-4 + 2)$.

☐ 40 ☐ −10 ☑ 24 ☐ −6
☐ none of these.

Figure 3: First page of questions, with "Begin Quiz" button and user-selections.

the question to its solution becomes accessible. This is done by hiding an opaque button which otherwise screens the active area of the link from receiving mouse-clicks.

Another novel aspect of the MacQTEX quiz system is the use of randomness to produce a slightly different quiz for each student access. Currently this is done using the *Mathematica*[3] [13] software, programmed to write a file of TEX definitions for each question. Other ways of doing this could be used; e.g. other software could be used, or sets of `\def`initions could be read from a file which has been generated in advance specially for this purpose.

Designing a new quiz

Constructing a new quiz is done at a MacQTEX quiz-site, using an HTML form, as shown in figure 8. Here an instructor may choose from pre-prepared question topics; currently there are 14 such types available, covering areas of basic mathematics.

For each topic, there are up to 6 actual question types. Both the number of topics and question types for a topic can be easily extended, though some knowledge of TEX or LATEX is needed to do this. When randomisation is required, then some knowledge of programming in *Mathematica* is also useful. For

[3] *Mathematica* is a trademark of Wolfram Research Limited.[13]

Preprint

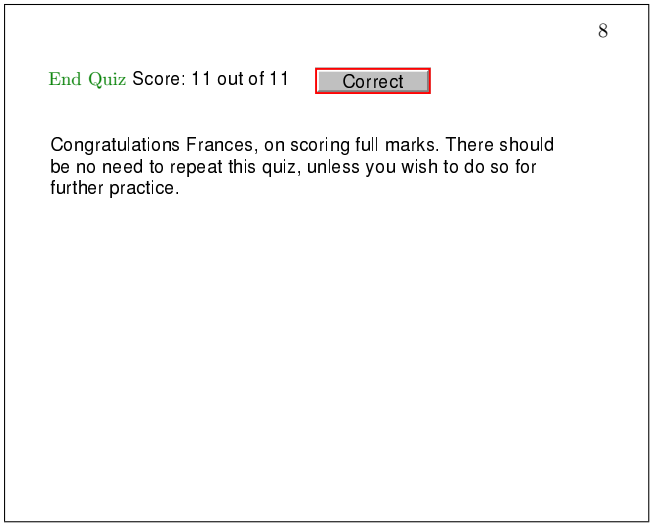


Figure 4: Last page of questions, after having selected the "End Quiz" button, showing the total score, and personalised confirmation message.

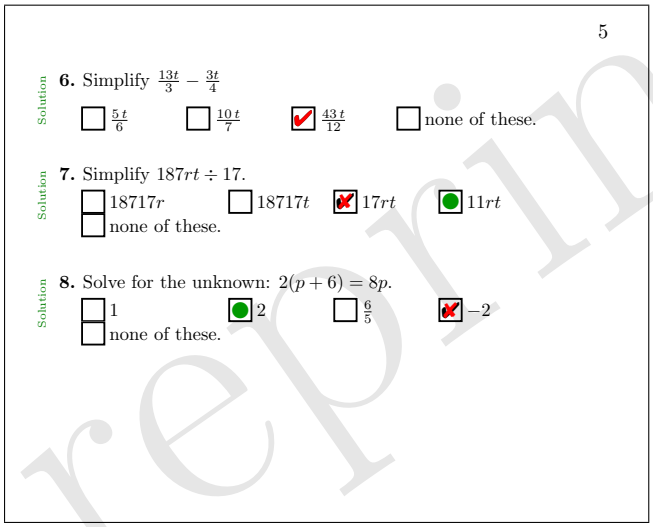


Figure 5: Embedded JavaScript [1, 9] is used to show the correct answer, when the student has made an incorrect choice. Also visible are buttons, previously hidden, which link to worked solutions.

Preprint

a successful quiz, it is necessary to generate plausible incorrect answers, as well as the correct answer. Generating these in *Mathematica* can be an interesting challenge. Sometimes it is necessary to discard random choices where an answer intended to be wrong actually agrees with the correct answer; that is, obtaining the correct answer by a completely fallacious method.

Figure 9 shows the work-flow for making a new quiz. This includes loops for producing example quizzes, and for editing of LaTeX sources for wording and/or layout. Only when the instructor is completely satisfied should a batch (e.g. 50) of quizzes be generated, and made available for student access.

LaTeX source code

Figure 10 shows the directory structure at a quiz site. It can be seen that there are many files with `.tex` suffix, which need to be read as part of a typesetting job. Reading all of these files in the correct order is essentially a bootstrap process, in which `\def`initions are made as required, before the next file is `\input`.

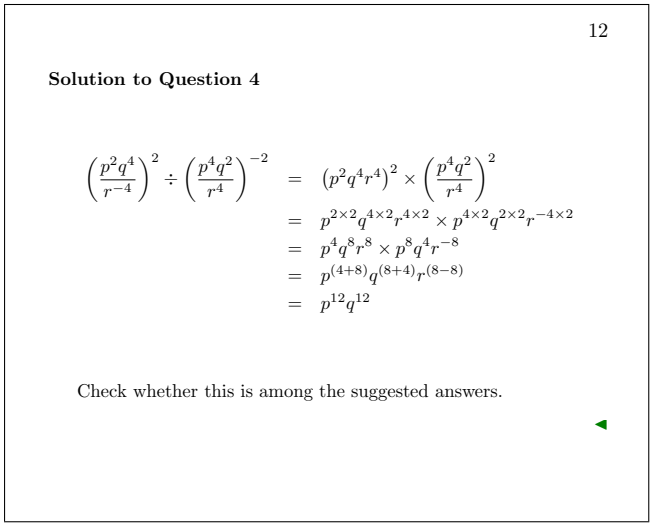


Figure 6: Worked solutions use properly typeset mathematics, as do the questions themselves. This one makes substantial use of mathematical symbols and equation alignments.

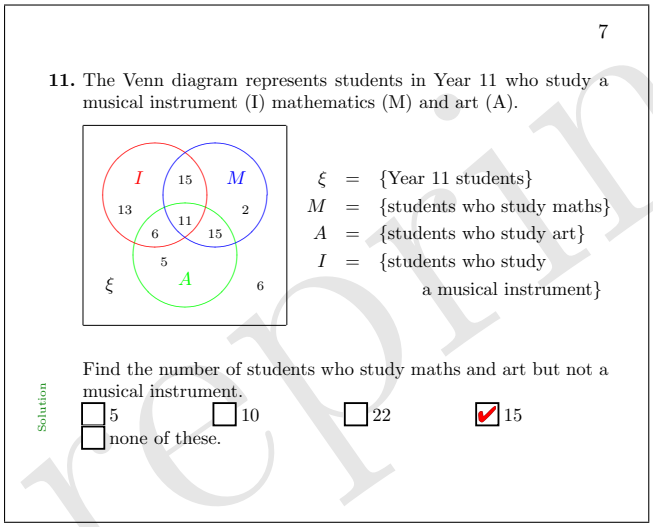


Figure 7: Elegant mathematical diagrams can also be used, both in the quiz questions and worked solutions.

For example, the main job for the quiz which was used for figures 2 to 6 uses a file having the name `MATH130quiz1.tex`, as follows:

```
\def\recipient{}
\def\defsdir{newquiz50/}
\def\texdir{../}
\nonstopmode
\catcode`\@=11
 \edef\eq@author{\recipient}
 \edef\eq@keywords{version 50}
\catcode`\@=12

\def\loginID{version 50}
\def\whichquiz{MATH130quiz1}
```

Preprint

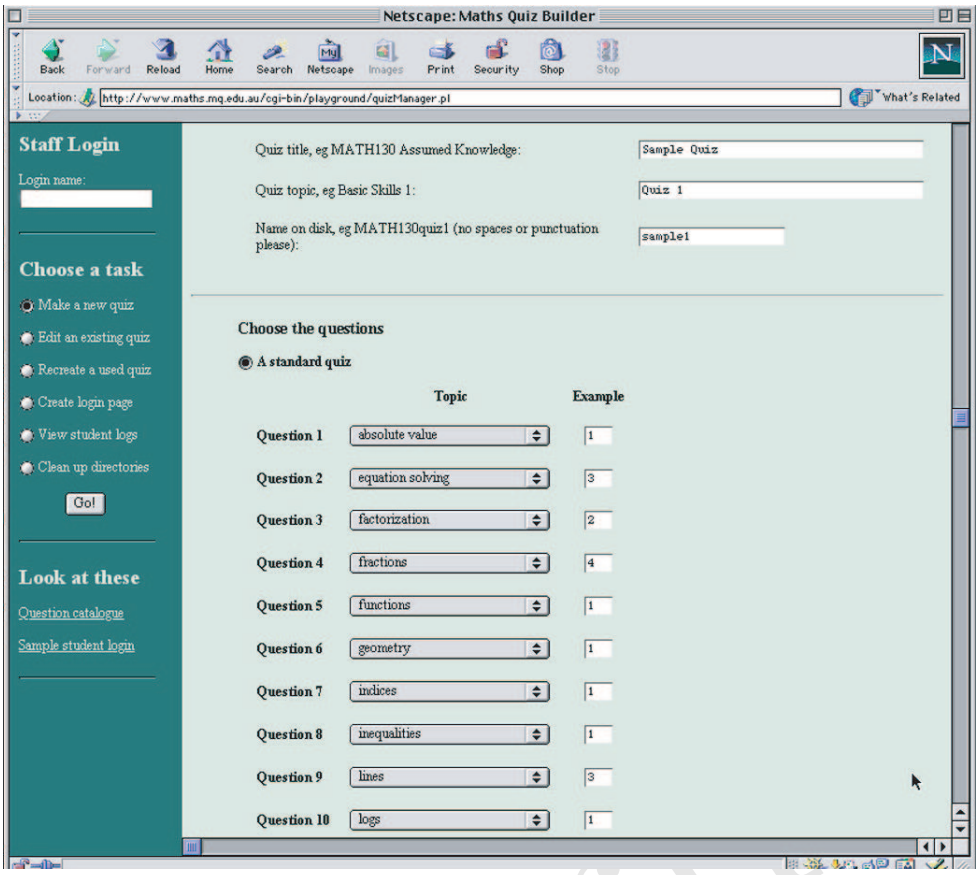


Figure 8: This web-page is used by instructors and course coordinators to design quizzes for the students to use. Questions can be chosen from pre-defined categories; each category has up to 6 choices of question.

```
\input \texdir user.tex
```

The number 50 that occurs here is because 50 instances of this quiz have been generated. For each instance the number would have been different.

The file `user.tex` is constant, for all quiz instances:

```
\def\author{Fran}
\def\imagedir{\texdir}
\input \texdir a.tex
\def\quizname{Basic Skills 1}
\input \texdir b.tex
\input \texdir bb.tex
\input \texdir c.tex
\input \texdir z.tex
```

Those files `a.tex`, `b.tex`, `bb.tex` and `z.tex` are constant for all instances of a particular quiz. Indeed `b.tex` and `c.tex` are created by a Perl [12] script, and hard-code variables such as the title of the quiz and its topic. It is `a.tex` which contains the `\documentclass` command, and loads the (modified) exerquiz package, as well as other standard LaTeX packages. Similarly, the `\end{document}` is in `z.tex`. These two files are simply copied from a global storage location.

Information for the opening page of a quiz is contained in `b.tex`. This file could well be edited to alter the instructions, or to convey other information about the quiz. The main information in `bb.tex` is the URL to which results submissions should be sent. Other specialised TeX definitions can be added here, when not suitable to be included in other packages.

Preprint

It is thus `c.tex` which controls input of the questions themselves, as follows:

```
%% You may edit this file to change the order of
% the questions, or adjust spacing and pagination.

\def\answerdir{\defsdir}
\def\CheckifGiven
 {Check whether this is among the suggested answers.}
\def\bfR{{\mathbf{R}}}
\def\bfZ{{\mathbf{Z}}}
\def\errOnSend
 {Your quiz results have not been received.}
%
\begin{quiz}*{\whichquiz}
%
\begin{questions}

\item \input{\texdir question1.tex}
\medskip
\vfil
\goodbreak
\vfilneg
\item \input{\texdir question2.tex}
\medskip
  ...
  ...
  ...
\goodbreak
\vfilneg
\item \input{\texdir question11.tex}
\medskip
\vfil
\goodbreak
\vfilneg
\end{questions}
\end{quiz}
%
%
\TextField[name=\whichquiz,width=1.25in,
  align=0,bordercolor={1 1 1},
  default=Score:,readonly=true]{}%
 \raisebox{3.5pt}\quad{\eqButton{\whichquiz}}
\therearequizsolutionstrue
\medskip
\TextField[name=progress,height=3cm,width=10cm,
  align=0,bordercolor={1 1 1},default=\errOnSend,
  multiline=true,readonly=true]{}
```

From this it can be seen that the source code for the questions themselves is contained in files named `question1.tex`, ..., `question11.tex`. This also contains the LaTeX source for the worked solution. One of these looks like:

```
\def\setupvalues{%
\def\fracsuma{1}\def\fracsumb{2}\def\fracsumc{5}
\def\fracsumd{9}\def\fracsump{9}\def\fracsumq{2}
\def\fracsumr{18}\def\fracsums{19}
\def\fracsumsign{+}
\def\ok{\frac{\fracsums}{\fracsumr}}
\def\wronga{3}\def\wrongb{\frac{6}{11}}}

\IfFileExists{\defsdir defs\thequestionno}
 {\input{\defsdir defs\thequestionno}}{\setupvalues}

Evaluate
${\frac{\fracsuma}{\fracsumb}}
  \fracsumsign {\frac{\fracsumc}{\fracsumd}}$


\RandomAnswers[123]{\answerdir answers\thequestionno}
\begin{answers}[\whichquiz:q\thequestionno]{5}
\theAnswersStream
%
\multicolumn{2}{l}{\Ans0 none of these. \hfil}
\end{answers}
%
\begin{solution}
\medskip
```

```
\def\setupvalues{%
\def\fracsuma{1}\def\fracsumb{2}\def\fracsumc{5}
\def\fracsumd{9}\def\fracsump{9}\def\fracsumq{2}
\def\fracsumr{18}\def\fracsums{19}
\def\fracsumsign{+}
\def\ok{\frac{\fracsums}{\fracsumr}}
\def\wronga{3}\def\wrongb{\frac{6}{11}}}

\addtocounter{solutionno}{1}

\IfFileExists{\defsdir defs\thesolutionno}
 {\input{\defsdir defs\thesolutionno}}{\setupvalues}

Using the common denominator \fracsumr
$$\frac{\fracsuma}{\fracsumb}
\fracsumsign \frac{\fracsumc}{\fracsumd}
 =\frac{\fracsuma\times\fracsump}{\fracsumr}
 \fracsumsign
  \frac{\fracsumc\times\fracsumq}{\fracsumr}
 =\ok.
$$
\medskip

\CheckifGiven
\end{solution}
\medskip
```

In the above TEX coding, the role of `\setupvalues` is to provide default definitions, just in case there is no file `defs\thequestionno` (that is `defs1`, `defs2`, ... according to the question number) within the `\defsdir` directory on the local file-system. If it does exist, (e.g., having been written by *Mathematica*[4], after performing calculations employing a random-number seed) then it should contain the necessary `\def`initions. Similarly there can be a file `\answerdir answers\thequestionno` to govern the order in which the answers are presented in the quiz question.

[4] *Mathematica* is a trademark of Wolfram Research Limited.[13]

Preprint

References

[1] Adobe Systems Inc.; "Acrobat Forms JavaScript Object Specification, Version 4.0"; Technical Note #5186; Revised: January 27, 1999.

[2] Adobe Systems Inc.; Acrobat Reader, viewer for PDF format [4] documents, available free of charge from http://www.adobe.com/.

[3] Adobe Systems Inc.; "FDF Toolkit Overview"; Technical Note #5194; Revised: August 10, 1999.

[4] Adobe Systems Inc.; "Portable Document Format, Reference Manual, Version 1.3"; March 11, 1999.

[5] Adobe Systems Inc.; "pdfmark Reference Manual"; Technical Note #5150; Adobe Developer Relations; Revised: March 4, 1999.

[6] Hàn, Thế Thành; pdfTEX, free software for generating documents in PDF format, based on the TEX typesetting system. Available for all computing platforms; see http://www.tug.org/applications/pdftex/.

[7] Lamport, Leslie; LATEX, a Document Preparation System. This is free software available for all computing platforms. Consult the TEX Users Group (TUG) website, at http://www.tug.org/.

[8] McKay, Wendy and Moore, Ross; "PDF presentations using the Marslide package." TEX Users Group 2001 Proceedings, (elsewhere in this volume).

[9] Netscape Communications Corporation; Nestcape JavaScript Reference, 1997; online at: http://developer.netscape.com/docs/manuals/communicator/jsref/toc.htm.

[10] Story, Donald; exerquiz & AcroTEX, packages for including special effects in PDF documents, using TEX and LATEX. Dept. of Mathematics and Computer Science, University of Akron. Software available online from http://www.math.uakron.edu/~dpstory/webeq.html.

[10] Story, Donald; "Techniques of Introducing Document-level JavaScript into a PDF file from a LATEX source." TEX Users Group 2001 Proceedings, (elsewhere in this volume).

[12] Wall, Larry; *Perl*, a general purpose scripting language for all computing platforms. This is Free Software, available from http://www.perl.com/.

[13] Wolfram Research Inc; *Mathematica*, a system for doing Mathematics by computer. Consult the website at http://www.wri.com/.

Preprint

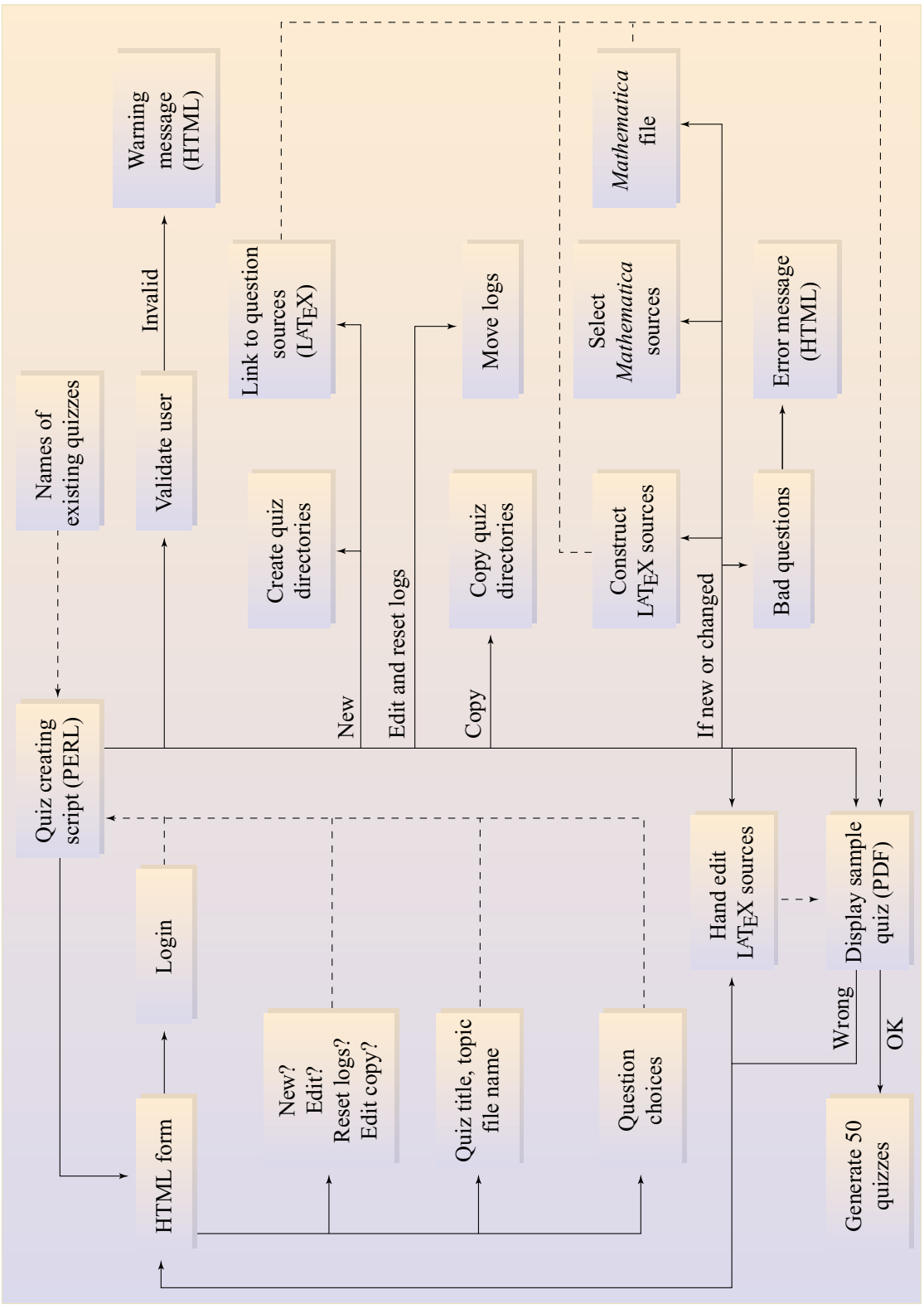


Figure 9: Work-flow for designing a new quiz, based on selections made from the HTML form shown in figure 8. Copies of existing files can be edited to taste. Constructing new question types requires more substantial editing, especially when randomisation is required.

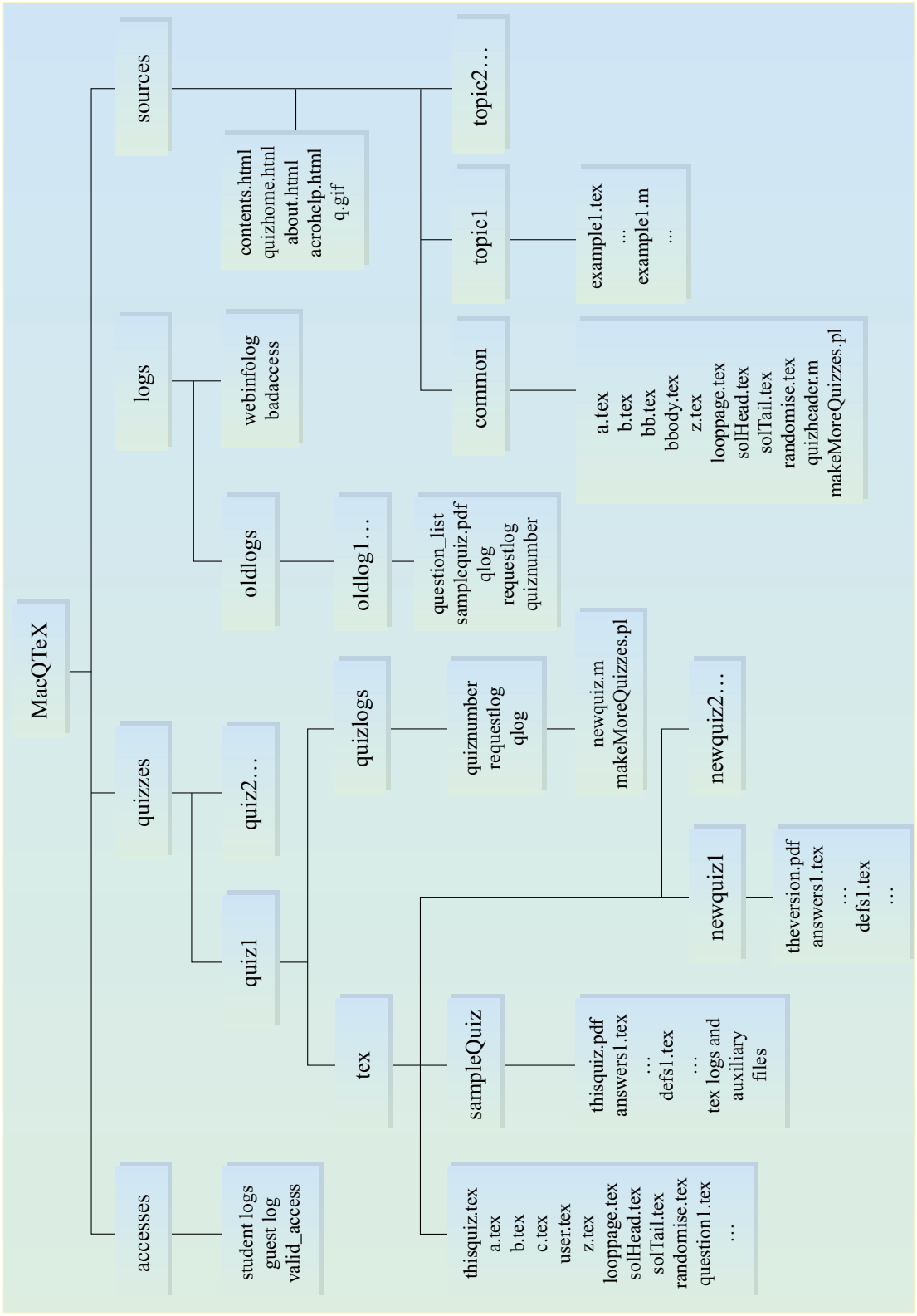


Figure 10: Directory structure of a quiz site, showing the locations of all the (LA)TEX input files for each question and quiz instance.

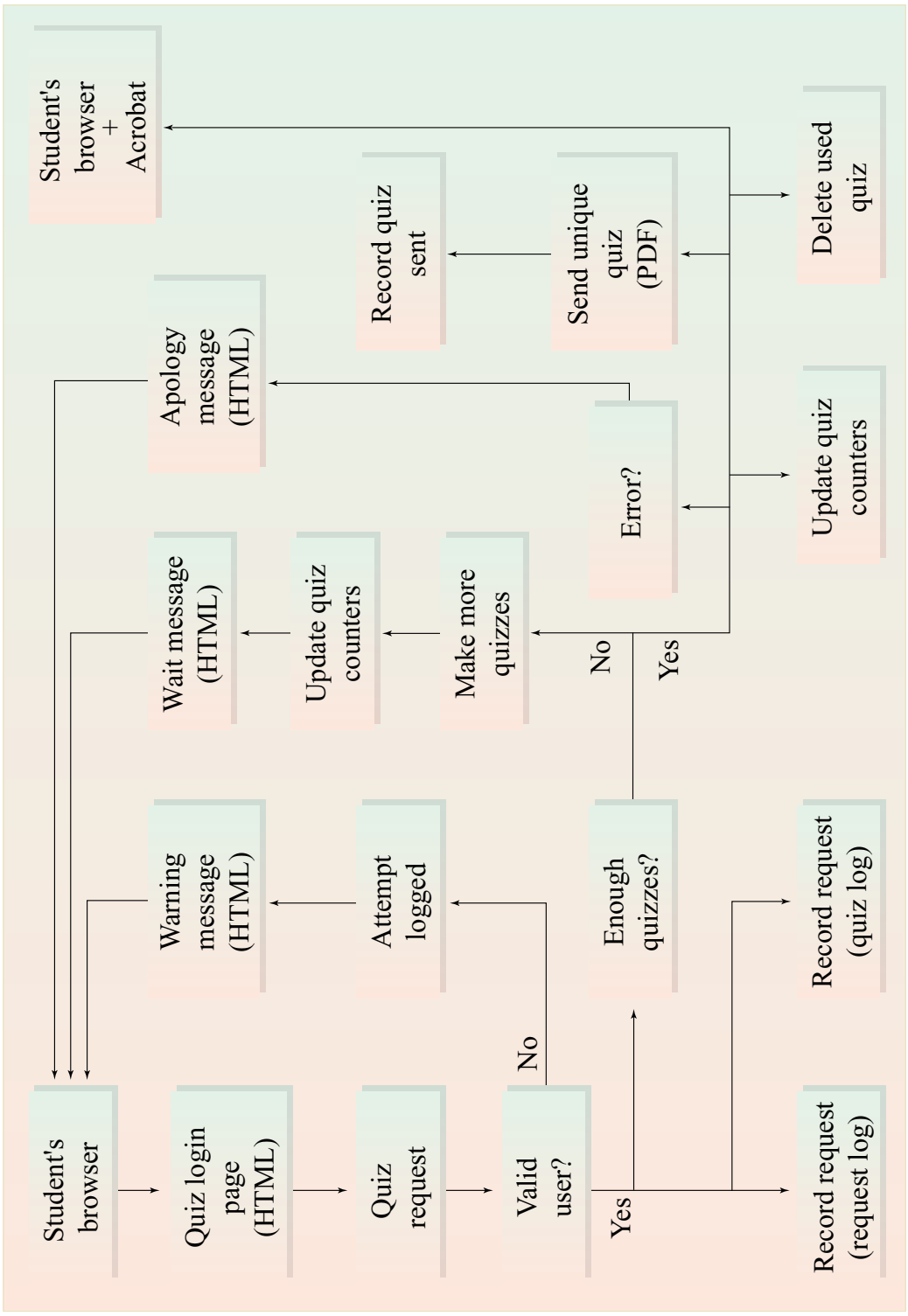
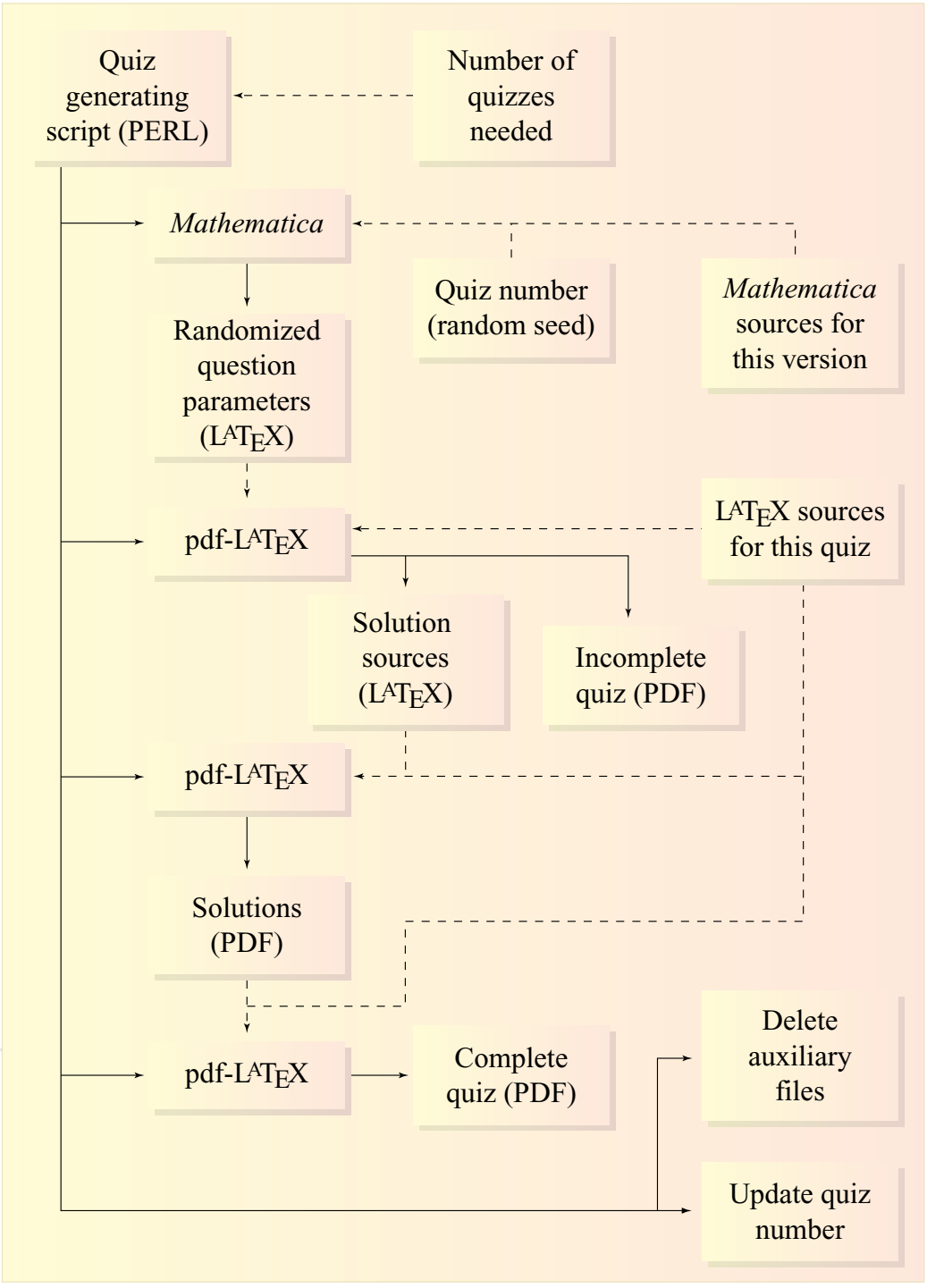


Figure 11: Detailed work-flow for sending a quiz to the student's browser. Only at the step "make more quizzes" does pdf-LaTeX come into play within this work-flow, as otherwise sufficiently many quiz documents have been typeset, awaiting requests for downloads. See figure 12 for the work-flow when it is necessary to generate quiz documents.



Generating new quizzes

Figure 12: Details of the sequence of calls to pdf-LaTeX for generating a quiz instance, along with its worked solutions. The first call causes a new document to be made containing the LaTeX coding for worked solutions. These are typeset at the 2nd call. Finally the 3rd call constructs a PDF document containing both the quiz questions and each worked solution, imported as a single graphic.

Figure 13: Presentation slides, mainly for use at education meetings. These give the rationale for using quizzes, as well as giving a rough representation of a student's interaction with the system. Also one slide has active hyperlinks to some actual web pages from where quizzes can be downloaded.

Figure 14: These presentation slides indicate how a random aspect is incorporated into the quizzes. Also shown are other aspects of generating quizzes, obtaining statistical information from the log-files, and general house-keeping duties that can be performed.

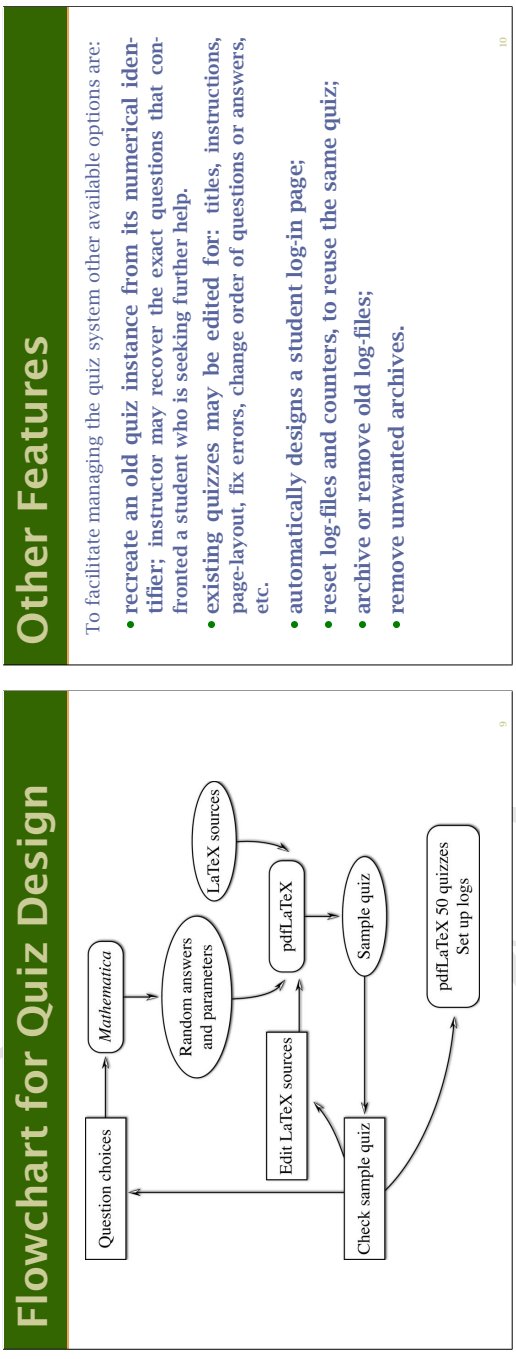


Figure 15: One slide here gives a rough view of the work-flow involved when preparing a set of quizzes for student use. A more detailed work-flow is giving in figure 11.

Preprint