

Spring 2022

## Canvas Autoquiz

Archit Jain

Follow this and additional works at: [https://scholarworks.sjsu.edu/etd\\_projects](https://scholarworks.sjsu.edu/etd_projects)



Part of the [Other Computer Sciences Commons](#), and the [Software Engineering Commons](#)

---

Canvas Autoquiz

A Project Report

Presented to

The Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Class

Spring-2022: CS 298

By

Archit Jain

May, 2022

© 2022

Archit Jain

ALL RIGHTS RESERVED

The Designated Project Committee Approves the Project Titled

Canvas Autoquiz

by

Archit Jain

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2022

Ben Reed

Thomas Austin

Robert Chun

Department of Computer Science

Department of Computer Science

Department of Computer Science

## **ABSTRACT**

Online learning management platforms such as Canvas are thriving and quickly replacing traditional classrooms, especially during these pandemic-struck times. As more and more quizzes are administered online, we need tools that make the quiz creation process easier and faster. Canvas Autoquiz is a command-line tool that allows instructors to automatically create and upload quizzes of varying difficulty levels. It also allows instructors to export quizzes from one LMS platform to another. This project explores the need, design, and implementation of the tool, and prospective future work.

## **ACKNOWLEDGMENT**

I would like to express my gratitude to Dr. Ben Reed for his direction, motivation, and support throughout this project. I feel fortunate as working with him had taught me the importance of building simple, yet powerful tools that improve user experience. He has encouraged critical thinking and kept me motivated throughout the project.

Next, I would like to thank the members of the MD2QTI team, Su Kim and Mingyun Kim for their efforts in the creation of the MD2QTI tool which acted as an inspiration for this project.

Next, I would also like to express my gratitude to the members of my defense committee, Prof. Thomas Austin, Prof. Robert Chun, and all the CS faculty for providing support over the two years of my degree.

Finally, I would like to thank my friends, peers, and family for their support and motivation.

## TABLE OF CONTENTS

List of Tables.....	v
List of Figures.....	vi
1. Introduction.....	1
2. History and Background .....	2
2.1 Canvas .....	2
2.1.1 Features of Canvas LMS.....	2
2.1.2 Canvas API Interface .....	6
2.1.3 Canvas Quiz import .....	9
2.2 MD2QTI.....	10
2.2.1 MD2QTI Usage.....	11
2.2.2 Supported Question Types .....	13
2.3 Markdown .....	17
2.4 QTI .....	19
3. Design Approach/Methodology .....	22
3.1 Historical Quiz Fetcher .....	23
3.2 Question Analyzer and Matcher .....	25
3.2.1 Difficulty Level Analysis .....	26
3.2.2 Question Similarity Checker.....	26
3.3 Database Folder Creator .....	28
3.3.1 Reason for using the file system to store the database.....	29
3.3.2 Current database design .....	30
3.3.3 Handling Question Variation.....	32
3.4 Database Folder Parser .....	32
3.5 Quiz Generator .....	33
3.5.1 Tagging and filtering .....	34
3.5.2 Selecting questions based on difficulty .....	36
3.5.3 Converting HTML to MD .....	36

- 3.6 Zip Creator .....37
- 4. Using Autoquiz .....38
  - 4.1 Download command.....38
  - 4.2 Generate command.....38
- 5. Setting Development Environment.....39
- 6. Difficulties and Challenges .....40
  - 6.1 Problem with setting up Canvas .....40
    - 6.1.1 Ruby setup using RVM .....40
    - 6.1.2 Installing Docker and Docker Compose .....41
    - 6.1.3 Installing YARN.....41
- 7. Future Work and Improvements .....42
  - 7.1 Question Tagging.....42
  - 7.2 Web Application .....42
- 8. Conclusion .....43
- References .....44



## LIST OF TABLES

Table 1: Internal data structures of Autoquiz .....	25
Table 2: Example of Levenshtein vs Hamming distance .....	28

## LIST OF FIGURES

Fig 1: Example Canvas dashboard.....	3
Fig 2: Generating Access Tokens in Canvas.....	8
Fig 3: Access token details in Canvas.....	8
Fig 4: Import course content in Canvas .....	9
Fig 5: Different import options in Canvas .....	10
Fig 6: example .md file that is required by MD2QTI .....	12
Fig 7: the content of manifest.xml in MD2QTI.....	13
Fig 8: Multiple Choice, True/False, and Multiple Answers supported by MD2QTI.....	14
Fig 9: Fill in the Blank, Multiple Blank/Dropdowns, and Matching questions supported by MD2QTI .....	16
Fig 10: Numerical and Formula questions supported by MD2QTI.....	17
Fig 11: File Upload and Essay questions supported by Md2QTI.....	17
Fig 12: Markdown example .....	18
Fig 13: Example of QTI 2.2 multiple choice questions .....	20
Fig 14: Autoquiz and the Ecosystem .....	22
Fig 15: Example of the config file of Autoquiz.....	24
Fig 16: Autoquiz fetching data from the Canvas .....	25
Fig 17: similar questions with the same group id .....	27
Fig 18: Example of the database folder for a practice course.....	31
Fig 19: Example question.html file for a question .....	31
Fig 20: Top k chosen values for a given difficult.....	36
Fig 21: converting HTML to markdown.....	37

## 1. INTRODUCTION

The onset of the global pandemic and various quarantine requirements have accelerated the transition of the classroom from a traditional in-person learning environment to a virtual classroom setting. This has led to several changes in the formal schooling system. One of the most notable changes has been the shift to online quizzes from paper-based quizzes. These online quizzes with advanced features such as proctoring have made the entire evaluation process easier, faster, and more efficient [21].

Canvas is one of the most popular virtual classroom tools today. It is a web-based learning management system that is primarily used by learning institutions, educators, and students for teaching and class administration [1]. It provides multiple features such as the ability to provide class materials, submit assignments, participate in discussions, and create/take quizzes.

Canvas is a great tool that solves a lot of problems; however, there are some limitations. For instance, although Canvas has support for the creation of quizzes and proctoring, the process of creating quizzes is manual and quite cumbersome. An instructor cannot create questions using scripts, find out the difficulty level of each question, and auto-generate questions based on tags and past quizzes. These limitations limit the overall effectiveness of the tool.

This paper presents Canvas Autoquiz, a command-line tool that aims to bridge this gap and make the quiz creation process easier, simpler and faster. This python-based tool aims to completely automate the quiz creation process and allows the instructors to download, analyze and store historical quizzes. The instructor can then use the tool to automatically create quizzes based on criteria such as question difficulty and tags. This tool uses the Canvas API interface to fetch data from Canvas and MD2QTI tool created by Su Kim [19] to upload the quiz to canvas.

An additional advantage of this tool is that it can allow Canvas quizzes to be imported to any other online-learning-based software that uses the QTI standard.

## **2. HISTORY AND BACKGROUND**

### **2.1 Canvas**

Canvas, originally named Instructure, was founded in 2008 and is an open-source web-based learning management system that is used by universities and other educational institutions to manage course materials [3]. Canvas allows for the creation of online courses where instructors can create online learning materials, and students can access these materials and utilize these to engage in learning. The software also tracks student skill development and achievements and can be used to improve the overall learning efficacy of a classroom. Canvas comes with several customizable course development and management tools, course and user analytics, as well as internal communication capabilities to provide a complete virtual classroom experience [1].

Canvas became the most popular learning management system (LMS), edging out the market leader Blackboard Inc. in 2018 in the United States and Canada [2]. Canvas owned 32% of the market in 2020 as it was further adopted by more American colleges and schools, while Blackboard Inc. owned 23% followed by Moodle at 22% [4].

#### 2.1.1 Features of Canvas LMS

The features offered by canvas can be divided into three main categories. Each of these parts is explored in detail:

1. Content creation
2. Assessment management
3. Support

Canvas has a great user interface that provides all its users with a dashboard, that contains a high-level overview of all the top courses, a to-do list, upcoming assignments, and a navigation menu that allows access to Canvas's core features [7].

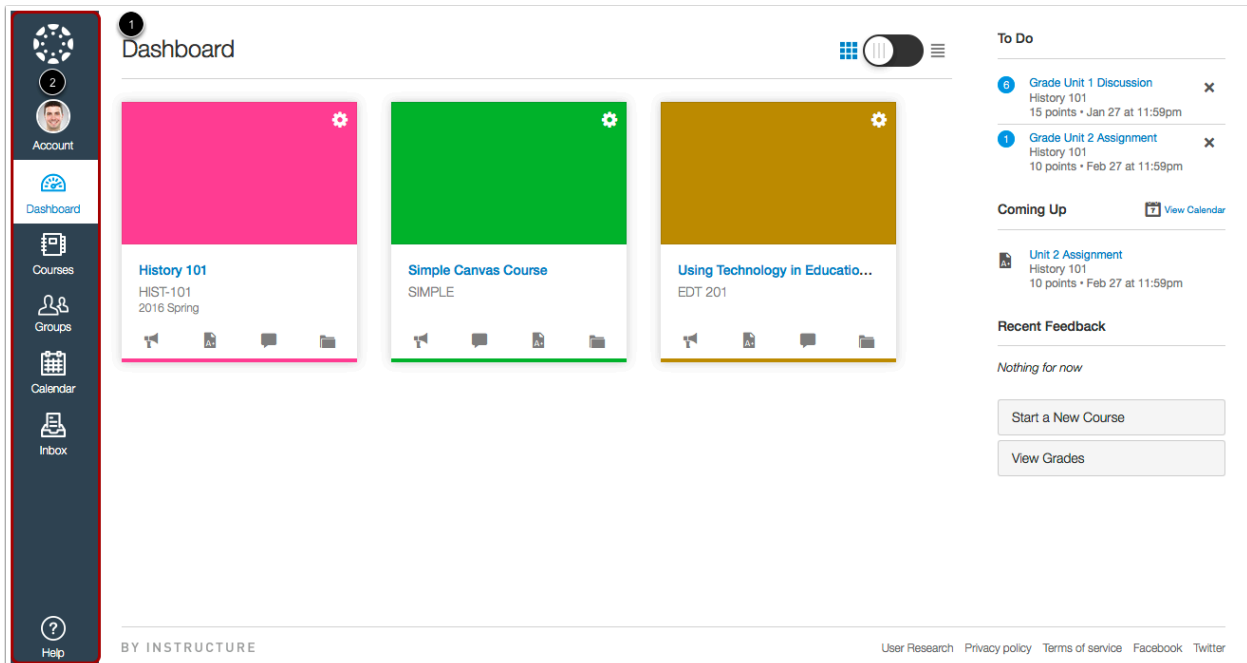


Fig 1: Example Canvas dashboard

## Course Creation

### *Canvas Commons*

Canvas adheres to the Common Cartridge requirements and has several features that aid in the construction and management of courses [6]. Courses can be created by users with administrative

and instruction privileges such as admins, designers, and instructors, and they can add assignments, modules, quizzes, comments, and online pages to a course. Instructors and admins can also share files, rubrics, and assignments with all the students in the course. Students can enroll in these courses, submit assignments, take exams/quizzes, and participate in online discussions with instructors and other students.

### *Modules*

Modules assist in the division of a course into multiple units, with each unit covering a specific topic of the course. The division of the course into smaller units helps students pace their learning. Instructors can also set prerequisites for modules such that they have to be taken in a specific order. Students do not see a course or unit until their prerequisites are completed. For example, a student must first complete the “understanding bits” unit before moving on to the “bit manipulation” unit.

### *Outcomes*

The outcome function is intended to assist institutions in determining a student's level of mastery of a particular subject area. This capability allows instructors and administrators to connect course content and student assessments to standard outcomes which are defined by the instructors themselves. This feature is especially useful for administrators who wish to compare the quality of education with other institutions.

## **Assessment Management**

### *Quizzes and Assignments*

Online quizzes allow students to take an exam/quiz or assignment whenever and wherever they want to take it. Quizzes can be scored immediately or graded at a later stage, can be timed or self-paced, and can be single or multiple attempts, with extensive descriptive answers or multiple choice questions with true or false answers, and many other types of questions.

Canvas assignments and quizzes can be used to test students' understanding and measure competency. The assignments page lists all of the assignments for a student, and instructors can also add detailed rubrics on the grading for each assignment.

### *Gradebook*

The Gradebook is a user interface that allows instructors, TAs, and administrators to quickly grade students' assignments. Once graded, the scores can be manually imported or exported in a CSV file, or be displayed automatically to students via Student Information System integration. The software remembers educators' preferences and habitual behaviors, allowing them to complete tasks with fewer clicks rather than picking, dragging, and dropping assignments repeatedly.

### *SpeedGrader*

SpeedGrader is a tool that allows educators to examine and annotate files. This user interface ensures that instructors are not required to download files and papers. Canvas supports a wide range of file types, including Google Docs, Microsoft Office, and PDFs. Educators can also make in-line annotations in the files, such as highlights, text strikeouts, and freehand drawings.

## **Support**

### *Learning Tools Interoperability*

Canvas uses Learning Tools Interoperability (LTI) [6], an IMS standard that helps third-party tools vendors to integrate their tools with canvas. Any compatible paid or free tools can be integrated with Canvas via LTI. Instructors can use these tools with any course in the current installation.

### *Reliable Connectivity*

Given that Canvas is deployed as a cloud-based solution in most institutions, high availability is paramount to its operation. Currently, Canvas LMS offers 99.99% uptime [6].

### *Prompt and Reliable Customer Support*

Instructure offers support via email and phone, and usually, trains staff dedicated to supporting each institution's installation.

### 2.1.2 Canvas API Interface

Canvas LMS includes a REST API for accessing and modifying data externally from the main application, using external programs and scripts. All registered students, faculty, graders, and admins can access the Canvas API simply by having a Canvas account [8]. The restrictions on the type of data that can be accessed through the API depend on one's role in Canvas. A student will have the same permissions when accessing Canvas through the API as when using the Canvas website; that means they can access their class schedule, due dates, and discussion posts, as well as look at their grades and assignment submissions.



Similarly, an instructor can access more information using the Canvas API such as the quiz details, the responses, grades of all the students, and many other features available to the instructor. Overall, while using the Canvas API, one can access all the data that is available on the user interface of the user.

### **Canvas Access Tokens**

To access the Canvas API, one must generate access tokens to grant access to Canvas resources. A user must obtain a token for access to the resources such as courses and quizzes using third-party applications.

The access tokens contain the same corresponding permissions as the user generating the token. The tokens are revoked if the corresponding Canvas account is deleted. The permissions of the tokens automatically adjust to the permissions associated with the user account. Any API requests to Canvas must include this token as a URL query parameter.

An instructor is only able to access the courses that they have access to and their corresponding quizzes using the access token they have generated. A student is unable to access any quizzes using an access code generated by them. An Administrator on the other hand can access all the courses and resources.

### *Generating Access Tokens*

Access tokens can be generated by going to *Account -> Settings -> Approved Integrations*

### Approved Integrations:

These are the third-party applications you have authorized to access the Canvas site on your behalf.

App	Purpose	Dates	
Inst-FS (Beta)		Expires: never Last Used: --	<a href="#">details</a>
Inst-FS (Beta)		Expires: never Last Used: --	<a href="#">details</a>
Training Portal		Expires: Jul 15 at 4:15pm Last Used: Jul 15 at 3:15pm	<a href="#">details</a>
Inst-FS (Beta)		Expires: never Last Used: --	<a href="#">details</a>
Canvas for Android	Nexus_7	Expires: never Last Used: Apr 28, 2014 at 3:44pm	<a href="#">details</a>
Canvas Teacher for Android	Pixel_3a	Expires: never Last Used: Mar 4 at 9:01am	<a href="#">details</a>
Canvas Parent for Android	Pixel_3a	Expires: never Last Used: Apr 27 at 1:25am	<a href="#">details</a>

[+ New Access Token](#)

Fig 2: Generating Access Tokens in Canvas

Click on *+New Access Token* and generate a new token. This will open a new window with token details [9].

### Access Token Details ×

Access tokens can be used to allow other applications to make API calls on your behalf. You can also generate access tokens and \*use the Canvas Open API\* to come up with your own integrations.

**1 Token:** 1081~prKjxoT8pyEo4DPBlgmTod2lhyVTNUj7aVhazDI3UKbwt3I4ha31c6YXgxCPn0xH  
**Copy this token down now.** Once you leave this page you won't be able to retrieve the full token anymore, you'll have to regenerate it to get a new value.

**App:** User-Generated

**Purpose:** Admin API

**Created:** Aug 14 at 7:43am

**Last Used:** --

**Expires:** Nov 30 at 12am

**2 Regenerate Token**

Fig 3: Access token details in Canvas

This token value is required when making a call to the canvas API. Autoquiz uses the canvas API to access the historical quizzes of an instructor and therefore needs a token generated by the instructor to access this data. More information on this can be found in the next section.

### 2.1.3 Canvas Quiz import

Canvas provides users such as instructors, graders, and admins to import course content into Canvas. The content which can be imported includes QTI quizzes, exported canvas courses, and Blackboard/Moodle courses. To import course content into a course, users can navigate to the Settings page of a course and choose the Import Course Content option on the right pane [10].

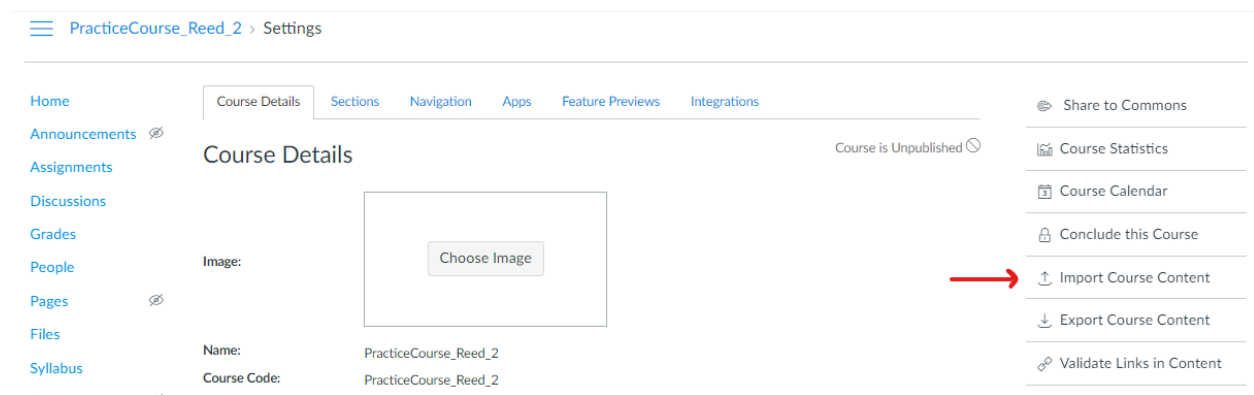


Fig 4: Import course content in Canvas

Given that Autoquiz needs to upload quizzes to canvas courses, we will focus on the QTI zip file option to import quizzes.

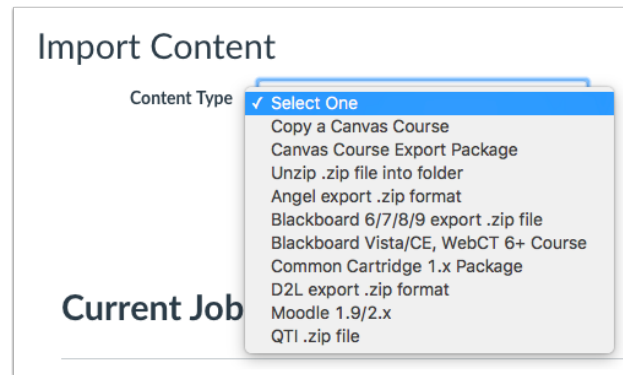


Fig 5: Different import options in Canvas

Upon successfully completion of a QTI zip file import, a quiz with the submitted contents will automatically be created and available in the quiz tab. A QTI import with invalid contents, such as unrecognizable XML bindings, will cause an error and no quiz will be generated.

## 2.2 MD2QTI

MD2QTI [19] is a Python-based software developed by Su Kim with contributions from Mingyun Kim and Archit Jain (myself). It is based on Text2QTI [20] developed by Geoffrey M. Poore, which allows text files with questions to be converted into QTI format. It supports the conversion of a variety of question types such as multiple-choice, short answers, essays, and many more. It converts Markdown question text files into QTI zip files that can be imported into multiple LMS platforms, including Canvas. MD2QTI builds upon Text2QTI and offers support for more question types and programmatically executable scripts [19].

MD2QTI supports some additional question types which were not supported by Text2QTI including Formula, Fill in Multiple Blanks, Multiple Dropdown, and Matching type questions. MD2QTI also adds a layer of consistency and removes ambiguity which was

prevalent in Text2QTI by adding special syntaxes such as @question title and @points which makes it easy for the user to create, understand and read the Markdown document.

The major advantage of using MD2QTI is the fact that it allows both manual and programmatic creation of questions. One can build a manual set of questions from which a specific number of questions can be selected at random. This allows the instructors to avoid repeating questions and enable them to tailor quizzes [19]. Another option is to write code that generates questions and adds them to the list of questions. The ability to write a script provides a quick and easy way to generate a large number of questions and ensure that question repetition is minimized.

M2QTI is simple and aims to minimize clicks in the Canvas interface. The user merely needs to type the questions in a specific format on an editor that is widely available on most PCs.

### 2.2.1 MD2QTI Usage

MD2QTI is a command-line tool that needs python 3 and mistletoe to run. To run MD2QTI, we need a .md file with correct syntax as seen in the figure below.

```

@quiz title: sample quiz
@quiz description: sample quiz to test md2qti
---
@title: Multiple choice
@points: 4
@type: multiple choice
@question:
Which one is LIFO?
@answer:
* Queue
* > Stack
* List
* Circular Queue
---
@title: Multiple answers
@points: 4
@type: multiple answers
@question:
Which one has O(n^2) time complexity?
@answer:
* > Insertion Sort
* > Bubble Sort
* Quick Sort
| * @feedback: please work
* Merge Sort
---
```

Fig 6: example .md file that is required by MD2QTI

We can then run the `md2qti` command with the sample file and it will return a zip file with the following content:

1. An `imsmanifest.xml` file
2. A `md2qti` assessment folder with `assessment_meta.xml` and `md2qti_assessment.xml`

The manifest is an XML document that describes the contents of the package. The manifest file provides details regarding metadata, organizations, resources, and any sub-manifests if needed [19].

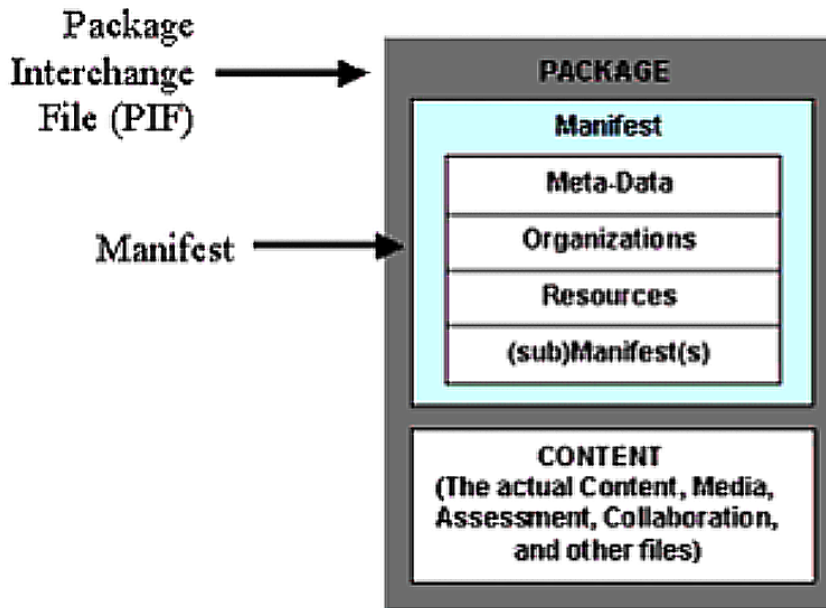


Fig 7: the content of manifest.xml in MD2QTI

The second part is the actual content of the quiz. Course materials, assessments, media, and other files can be found in this folder.

### 2.2.2 Supported Question Types

Currently, users can create eleven types of questions that are supported by Canvas. Autoquiz takes the questions in the internal database and converts them into the format below (right side of the images). This format is recognized by MD2QTI, which then converts it into the format on the left side of the images below.

The first group of questions is Multiple Choice, True/False, and Multiple Answers. These questions are logical-based questions where the user must select one or multiple answers from a pre-defined set of choices. These questions can be automatically graded.

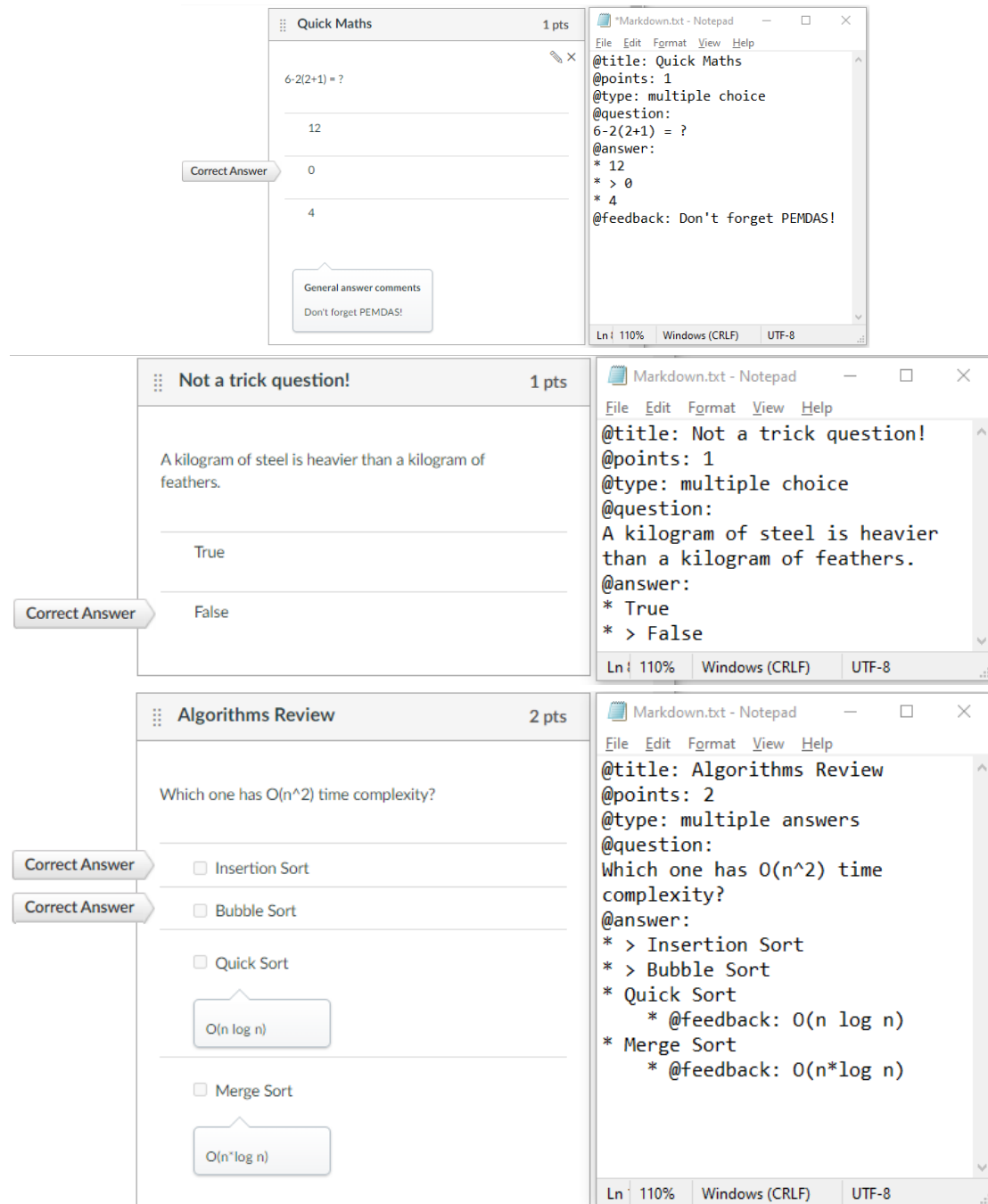


Fig 8: Multiple Choice, True/False, and Multiple Answers supported by MD2QTI

The next group of questions is Fill in the Blank, Fill in Multiple Blank, Multiple Dropdowns, and Matching. The idea behind this group is that the user must match the correct answer from



multiple choices. In some cases, the correct answer is based on equality or pattern matching.

These questions can be automatically graded.

The image displays three examples of Canvas AutoQuiz questions and their corresponding Markdown code. Each example consists of a question interface on the left and a Notepad window showing the Markdown code on the right.

**Example 1: A simple joke (4 pts)**

Question: What do you call a fish without an eye?

Correct Answers: fsh, FSH, Fsh

Markdown Code:

```
@title: A simple joke
@points: 4
@type: blank
@question:
What do you call a fish without
an eye?
@answer:
* > fsh
* > FSH
* > Fsh
```

**Example 2: Number Systems (9 pts)**

Question: 0x is a prefix for [box1]. 0b is a prefix for [box2].

Show Answers for: box2

Correct Answer: binary

Correct Answer: Binary

Markdown Code:

```
@title: Number Systems
@points: 9
@type: multiple blanks
@question:
0x is a prefix for [box1].
0b is a prefix for [box2].
@answer:
+ box1
  * > hexadecimal
  * > Hexadecimal
+ box2
  * > binary
  * > Binary
```

**Example 3: Map check! (2 pts)**

Question: [box1] is a city, [box2] is a state.

Show Answers for: box1

Correct Answer: San Jose

Correct Answer: Texas

Correct Answer: Arizona

Markdown Code:

```
@title: Map check!
@points: 2
@type: multiple dropdown
@question:
[box1] is a city, [box2] is a
state.
@answer:
+ box1
  * > San Jose
  * > Texas
  * > Arizona
+ box2
  * > Los Angeles
  * > Seattle
  * > California
```

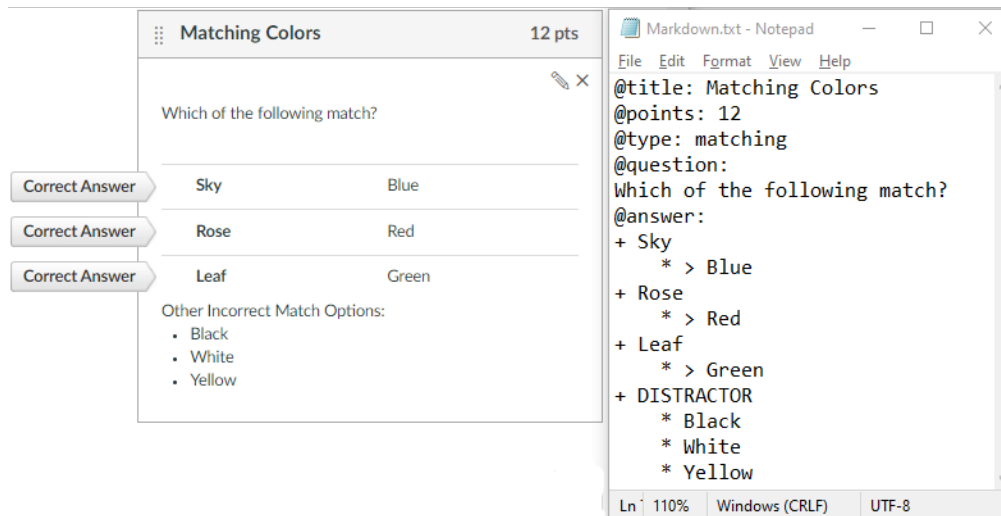


Fig 9: Fill in the Blank, Multiple Blank/Dropdowns, and Matching questions supported by MD2QTI

The third group of questions is Numerical and number-based Formula questions. Students must calculate and answer with just numbers. These questions are often with calculated precision or specified decimal places. These questions can be automatically graded.

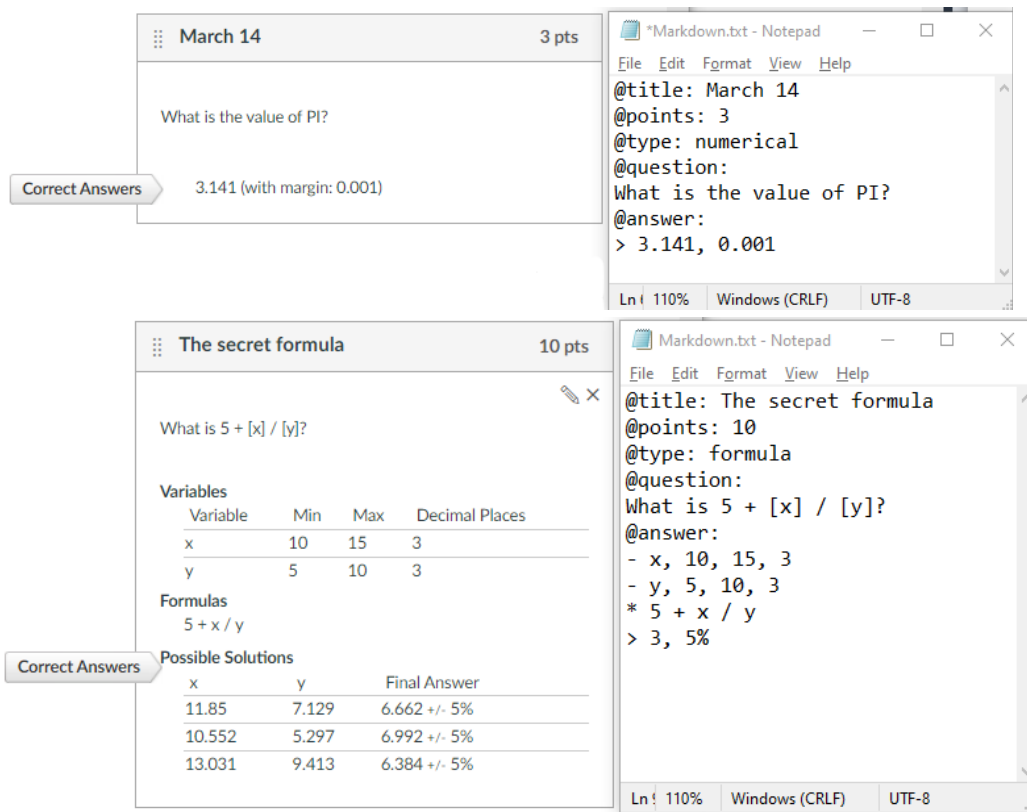


Fig 10: Numerical and Formula questions supported by MD2QTI

The final group of questions is File Upload and Essay. For these questions, the user must manually upload a file or type out a response. These questions cannot be automatically graded and the instructor needs to manually grade them.

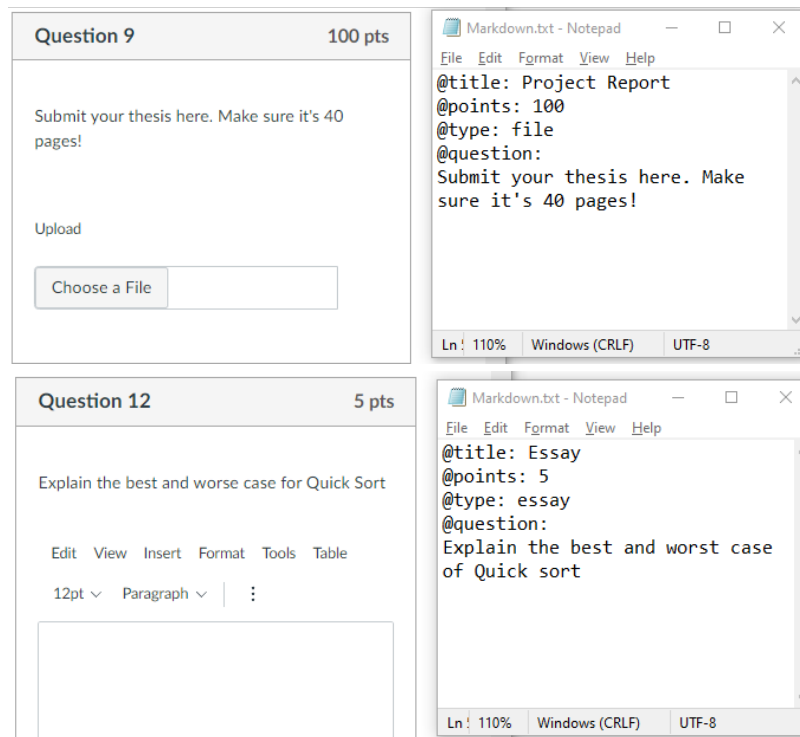


Fig 11: File Upload and Essay questions supported by Md2QTI

## 2.3 Markdown

Markdown is a simple markup language that can be used to add formatting to plaintext text documents. Markdown, which was created by John Gruber in 2004, is currently one of the most widely used markup languages in the world [11].

Using Markdown is not the same as using a text or document editor. In a program like Microsoft Word, one may format words and sentences by clicking buttons, and the changes are immediately visible. That isn't the case with Markdown. When creating a Markdown-formatted document, one must use Markdown syntax to designate which words and phrases should be formatted differently.

To indicate a heading, for example, place a number sign before it (e.g., # Heading One). Alternatively, you can make a phrase bold by placing two asterisks before and after it (for example, **this text is bold**) [12].

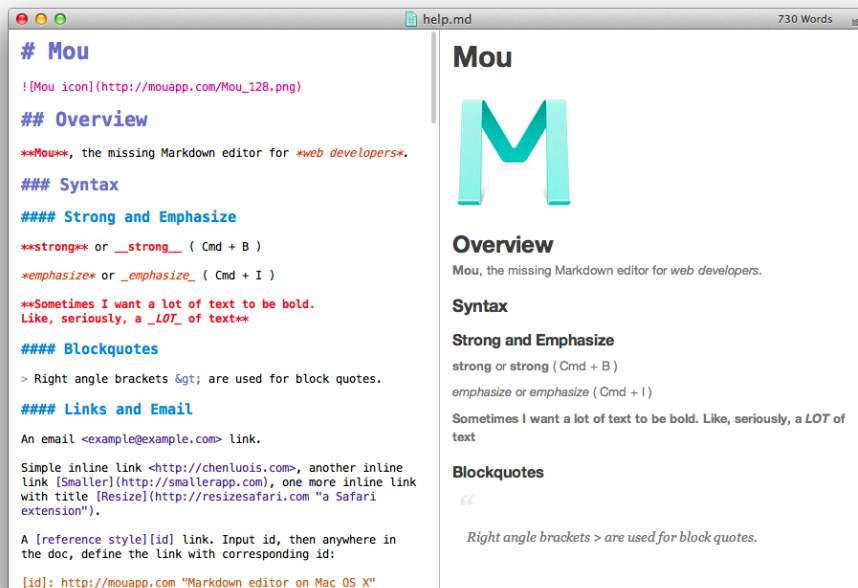


Fig 12: Markdown example

Everything can be written in Markdown. Websites, documents, notes, books, presentations, email messages, and technical documentation are all created with it. Furthermore,

Markdown is a platform-agnostic portable text editor and therefore virtually any application may open files containing Markdown-formatted text.

Markdown is a future-proof language. One will be able to view the Markdown-formatted material using any text editing application even if the application one is using stops working in the future. Markdown is all over the place and is supported by several desktop and web-based apps, including Reddit and GitHub.

## 2.4 QTI

The IMS Question and Test Interoperability specification (QTI) [13] establishes a standard format for presenting assessment content and results, facilitating the transfer of this information between authoring and delivery systems, repositories, and other learning management systems. It enables assessment materials to be written and delivered on a variety of platforms. As a result, it's built to make system interoperability easier.

The standard includes an XML data binding that effectively defines a language for exchanging questions and other assessment material, as well as a data model that describes the structure of questions, assessments, and results from questions and assessments. The XML binding is commonly used by publishers for transmitting questions between different publishing programs. The elements of the standard dealing with assessment and results are less commonly used.

The IMS Global Learning Partnership (IMS GLC), which is an industrial and academic consortium that produces compatible learning technology requirements, has developed QTI to help users avoid losing or having to retype questions as technology changes. It can take a long

time to develop and validate appropriate questions, thus it's preferable to be able to do so in a platform and technology-agnostic style. Currently, Canvas supports QTI versions 1.2 and 2.1.

QTI version 1.0 was initially built on proprietary Questions Markup Vocabulary (QML) language; however, the language has grown over time and can now describe practically any acceptable question.

QTI version 2.0 was released in 2005 and solely addressed the item level of the specification (i.e., each question). In 2005, a draft version of Version 2.1 was released, which addressed the organization of tests and outcomes [14].

The most recent version is 2.2, which was released in 2015 after two minor changes, 2.2.1 and 2.2.2, the most recent of which occurred in November 2017.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This example adapted from the PET Handbook, copyright University of Cambridge ESOL Examinations -->
<assessmentItem xmlns="http://www.imslobal.org/xsd/imsqti_v2p2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.imslobal.org/xsd/imsqti_v2p2 http://www.imslobal.org/xsd/qti/qtiv2p2/imsqti_v2p2p2.xsd"
  identifier="choice" title="Unattended Luggage" adaptive="false" timeDependent="false">
  <responseDeclaration identifier="RESPONSE" cardinality="single" baseType="identifier">
    <correctResponse>
      <value>ChoiceA</value>
    </correctResponse>
  </responseDeclaration>
  <outcomeDeclaration identifier="SCORE" cardinality="single" baseType="float">
    <defaultValue>
      <value>0</value>
    </defaultValue>
  </outcomeDeclaration>
  <itemBody>
    <p>Look at the text in the picture.</p>
    <p>
      
    </p>
    <choiceInteraction responseIdentifier="RESPONSE" shuffle="false" maxChoices="1">
      <prompt>What does it say?</prompt>
      <simpleChoice identifier="ChoiceA">You must stay with your luggage at all times.</simpleChoice>
      <simpleChoice identifier="ChoiceB">Do not let someone else look after your luggage.</simpleChoice>
      <simpleChoice identifier="ChoiceC">Remember your luggage when you leave.</simpleChoice>
    </choiceInteraction>
  </itemBody>
  <responseProcessing
    template="http://www.imslobal.org/question/qti_v2p2/rtemplates/match_correct"/>
</assessmentItem>
```

Fig 13: Example of QTI 2.2 multiple choice questions

Version 2.2 refined and updated the Version 2.1 core specification's connection with W3C standards such as HTML5, SSML, PLS, CSS, ARIA, and MathML, and made other minor adjustments [14].

### 3. DESIGN APPROACH/METHODOLOGY

Autoquiz fits in the spectrum of tools designed to help instructors make quizzes much faster. It interacts with canvas and MD2QTI as shown in the figure below.

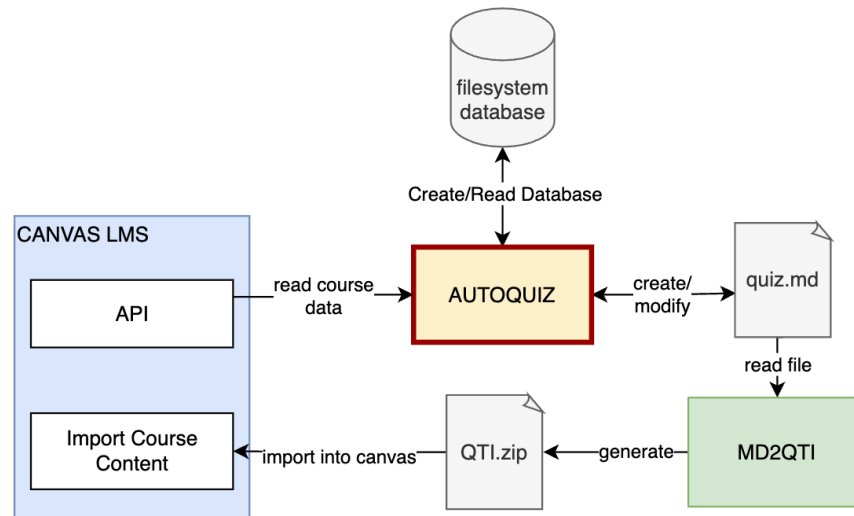


Fig 14: Autoquiz and the Ecosystem

It can be seen that Autoquiz acts as an intermediate to extract the quiz data from Canvas and stores it in a local filesystem-based database. It then generates a quiz.md file with the relevant questions which is utilized by MD2QTI to upload the generated quiz to Canvas.

The project follows the Unix design philosophy, which is a set of cultural norms and philosophical approaches to minimalist, modular software development. The aim is to build simple, short, clear, modular, and extensible code with a focus on maintainability and easy enhancements. The idea is to split each part of the program to do one thing well [5]. The output of each of the parts of one module becomes the input for the next module. This allows each module to be used separately, easily enhanced, or replaced and makes the tool much easier to



understand. We will dive into the reasoning behind the division in each of the individual modules. The implementation of the project can be divided into multiple modules:

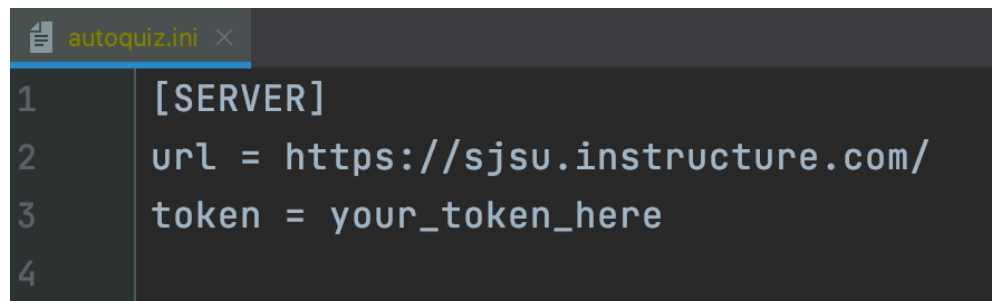
1. Historical Quiz Fetcher
2. Question Analyzer and Matcher
3. Database Folder Creator
4. Database Folder Parser
5. Quiz Generator
6. Zip Creator

### **3.1 Historical Quiz Fetcher**

The project aims to automate the quiz creation process for the instructors. To be able to create a new quiz, the tool needs to have an existing database of questions from which it can filter out questions based on the needs of the instructor. Now, the instructor can create the database from scratch; however, we already have a pool of questions stored in canvas which makes it very easy for instructors to use an existing pool of questions. These questions are stored as all the past quizzes that an instructor might have conducted on canvas. Moreover, instructors can collaborate and share the database and the relevant question statistics with other instructors to create a very large question bank.

This module focuses on fetching historical quiz data from Canvas by using Canvas API. The Canvas API can be accessed using the Canvas Access Token (see Canvas under History and Background section). A historical quiz is classified as all the old quizzes that the instructor might have taken over the multiple courses in canvas. The course might be active or inactive on canvas.

An instructor must generate a new access token which must be fed into the config file of the tool before it can be used. The instructor must ensure that the access token is not expired, otherwise, the tool will be unable to connect to Canvas. In case of an expired token, Canvas Autoquiz will throw an error and request a new token. The config file also needs the API end-point through which the API will be called. Each instance of canvas deployed by an institution will have a different API end-point through which we can call the API and access the data stored by canvas. An example of the config file (named autoquiz.ini) can be seen below.

A screenshot of a code editor window titled 'autoquiz.ini'. The window contains four lines of text: line 1 is '[SERVER]', line 2 is 'url = https://sjsu.instructure.com/', line 3 is 'token = your\_token\_here', and line 4 is empty. The text is in a light gray font on a dark background.

```
1 [SERVER]
2 url = https://sjsu.instructure.com/
3 token = your_token_here
4
```

Fig 15: Example of the config file of Autoquiz

This module uses the “canvasapi” python module [22] by ucopen which is a Python API wrapper for Instructure's Canvas LMS. This made the development of the Canvas Autoquiz tool much easier as the wrapper handled all the API calls.

To be able to download all the historical quizzes, the Canvas Autoquiz tool iterates through all the courses of the instructor. It then iterates through all the quizzes in the course and downloads statistics and other metadata for each question in the quiz. All this information is stored as an object of the corresponding data structure in the memory.

```

INFO: connected to canvas as Archit Jain (4474687)
found 1 courses.

processing course: PracticeCourse_Reed_2
  processing quiz: Test Quiz
    processing quiz: practice exam 1 with setup validation- Requires Respondus LockDown Browser
  processing quiz: exam1- Requires Respondus LockDown Browser
  processing quiz: exam 2- Requires Respondus LockDown Browser
  processing quiz: Test Quiz
  processing quiz: Practice Exam 2- Requires Respondus LockDown Browser

```

Fig 16: Autoquiz fetching data from the Canvas

The internal objects of the canvas data are stored in data structures of three types:

Type	Description
Course	One course which contains a list of quizzes and the corresponding course data.
Quiz	One quiz which contains a list of questions and the corresponding course data.
Question	One question which contains a list of variations of the particular questions and question statistics

Table 1: Internal data structures of Autoquiz

### 3.2 Question Analyzer and Matcher

Before we dump all the fetched data into a database, we need to analyze the questions to find question variations and combine similar questions. This module focuses on analyzing the fetched quizzes to find out the difficulty level of each question. Furthermore, we try to match similar questions and combine their statistics to allow the instructor to view the overall statistic of a particular question type. This functionality is especially useful in the case of script-generated questions where the only difference between the questions is some numerical digits.

### 3.2.1 Difficulty Level Analysis

Canvas stores statistics for each question in a quiz. Data stored includes stats such as the number of students that attempted the question, correct responses, and many others. Another field that is stored is the difficulty index. The difficulty index is calculated by dividing the total number of correct responses divided by the total number of responses.

In the case of programmatically generated questions, or question groups with many questions and fewer picks, there might be questions that were not attempted by students and therefore might not have any associated statistics. In this case, the difficulty level cannot be calculated. In this case, it is important to match the questions with similar questions and to combine their statistics.

### 3.2.2 Question Similarity Checker

A lot of programmatically generated questions are quite similar on canvas. This means that they test the same concept but use a different value to do the same. In Autoquiz, we aim to group these questions together so that we can avoid question repetition in the final generated quiz.

To check the similarity of questions, two concepts are used. These are:

1. Group id
2. Levenshtein distances

#### *Group id*

All the questions created using the same group will be treated as similar questions. This is because similar questions are grouped in the quiz creation process. This can be seen in the figure

below where two questions with similar “question\_text” have the same group id as they were created using a script.

```
"question_name": "Question",  
"question_text": "<p>the decimal number 327 can be represented the following ways:",  
"question_type": "fill_in_multiple_blanks_question",  
"quiz_group_id": 895942,  
"quiz_id": 1574894,
```

```
"question_name": "Question",  
"question_text": "<p>the decimal number 326 can be represented the following ways:",  
"question_type": "fill_in_multiple_blanks_question",  
"quiz_group_id": 895942,  
"quiz_id": 1574894,
```

Fig 17: similar questions with the same group id

### *Levenshtein distances*

We can make the following assumptions about similar questions from a quiz.

1. The questions will be of similar length
2. The questions will alter for each mostly on numerical values
3. The order of words will not differ too much

This is based on the observation of questions from the past quizzes in Prof. Ben Reed’s CS 149 Operating System courses.

The Levenshtein distance, commonly referred to as edit distance, is a string metric used to compare two sequences. The Levenshtein distance between two words is the smallest number of single-character modifications (insertions, deletions, or substitutions) required to transform one word into the other [15]. For example, the Levenshtein distance between "kitty" and "sitting" is 3, as it would require three modifications to turn “kitty” into “sitting”.

Another popular metric for measuring string similarity is the Hamming distance. Hamming distance is the number of locations at which the matching symbols are different between two strings of equal length. In other words, it calculates the smallest number of substitutions required to transform one string into another or the smallest number of errors that may have resulted in the transformation of one string into the other [16]. The Hamming distance is one of numerous string metrics for determining the edit distance between two sequences in a more broad sense.

In our case, it seems more appropriate to use the Levenshtein distances over hamming distance as we expect only some parts of the question to be different, and the overall majority of the question will be the same.

*For Example:*

Question 1: Convert 100101011 from binary to hexadecimal.

Question 2: Convert 1011010 from binary to hexadecimal.

Type	Distance	Reason	Calculation
Levenshtein	3	3 modification (1 substitution, 2 insertions)	1011010 -> 1001010 -> 1001010 <u>1</u> -> 1001010 <u>11</u>
Hamming	29	29 different positions where bits are different	Convert 100101011 from binary to hexadecimal. Convert 1011010 from binary to hexadecimal.

Table 2: Example of Levenshtein vs Hamming distance

The calculation of the Levenshtein distance is taken care of by the Levenshtein Python module [23] which can be downloaded for the pip package manager.

### 3.3 Database Folder Creator

Once similar questions have been grouped together, we can finally move all the data from the memory to a database. This module focuses on creating a system that stores the instructor's

downloaded quizzes on their system in a database folder. This database is a simple folder structure containing data and metadata for all the courses, quizzes, and questions.

### 3.3.1 Reason for using the file system to store the database

Initially, it was decided that a SQL or NoSQL database such as MySQL or MongoDB would be used to store the course, quiz, and question information. However, this idea was dropped after consultation with the project advisor.

SQL and NoSQL databases would have allowed for much faster and simpler code as most databases contain python modules that simplify the task of storing and retrieving data. This module and the Database parser module would not have been required in case this strategy was adopted. However, there are several downsides to using these databases. Firstly, the user would have to download and set up a SQL or NoSQL database. This would have limited the tool to instructors who have development knowledge. Furthermore, it would make the tool quite restricted for future development as a new developer would have to go through the code or the documentation to understand the structure of the data. This goes against the Unix design philosophy and would have resulted in a suboptimal tool.

For the above reasons, a file-based database is a more appropriate design. This is because it is not expected that the past quizzes would be greater than the available memory on the instructor's machine. This means that the tool would only need to write and read the historical quizzes to the disk once and then can read all the available data from memory. This will be much faster than using a SQL or NoSQL database. Furthermore, there will be no overhead from an external server and it would be very easy for any other developer to modify the tool for their use case.

Moreover, the folder can be easily moved from one machine to another without any specialized tools or skillset. It is also quite easy for a non-technical user to understand the quiz structure and make changes to the questions without understanding the code. This module combined with the Historical Quiz Fetcher module can serve as an archival tool that can be used to backup Canvas Quiz data into a human-understandable format.

### 3.3.2 Current database design

The design of the current database can be seen below. The main database folder contains a sub-folder which the name of the folder corresponding to the name of the courses. Each course will have multiple quiz sub-folders and a metadata.json file with the course metadata. Each quiz will have multiple question sub-folders and a metadata.json file with the quiz metadata.

```
Database
  \-> Course1
    \-> Quiz1
      \-> Question1
        \-> question0.html
        \-> metadata0.json
      \-> Question2
        \-> question0.html
        \-> metadata0.json
        \-> question1.html
        \-> metadata1.json
      \-> Question3
      \-> Question4
      \-> metadata.json
    \-> Quiz2
    \-> Quiz3
    \-> metadata.json
  \-> Course2
  \-> Course3
```

Each question will have a question.html and a metadata.json file. The HTML file is used so the user can easily view the question including the question images and the question text just as it would appear in the final canvas quiz.



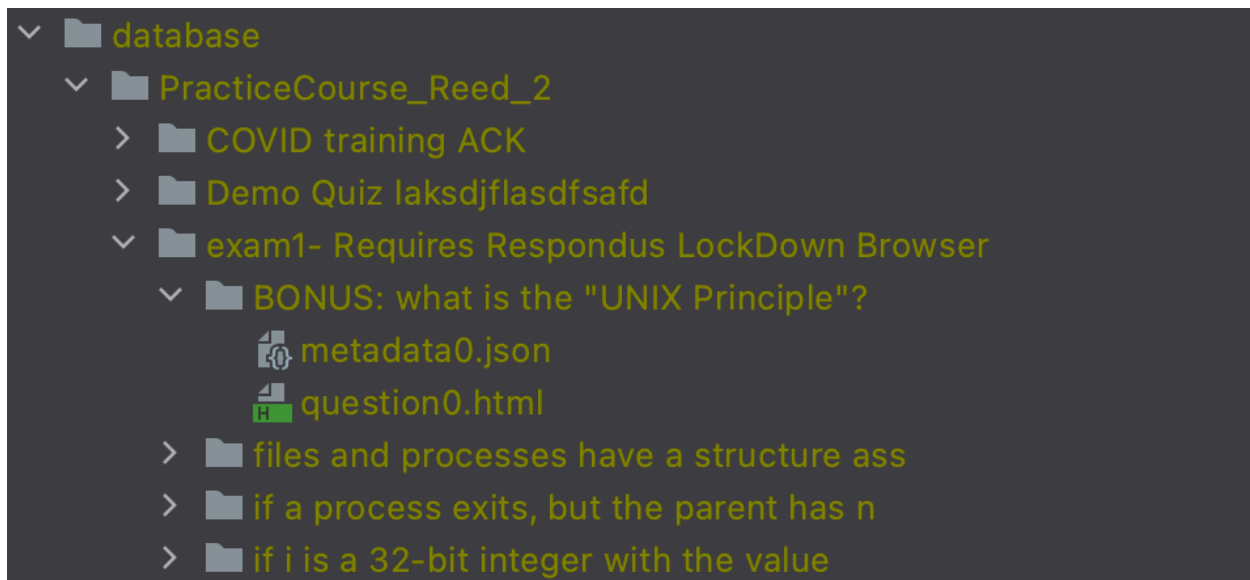


Fig 18: Example of the database folder for a practice course

```

<p>put a number from 1-4 next to each stage of starting an operating system.</p>
<p>[1] start init</p>
<p>[2] execute initial BIOS or EFI code</p>
<p>[3] bootloader</p>
<p>[4] load and execute the kernel</p>

```

put a number from 1-4 next to each stage of starting an operating system.

[1] start init

[2] execute initial BIOS or EFI code

[3] bootloader

[4] load and execute the kernel

Fig 19: Example question.html file for a question

Having this structure allows any user to easily look at and modify the content of the quizzes without knowledge of the working of the tool.

### 3.3.3 Handling Question Variation

In the case of questions that have been grouped together due to their similarity, only one of the question folders is created. This folder is different from single questions in the way that each variation of the question has its question.html and metadata.json file. The format of a single question is given below.

```
\-> Question1
    \-> question0.html
    \-> metadata0.json
```

The format of a question with multiple variations is given below. This ensures that the user can see all the variations of the question.

```
\-> Question2
    \-> question0.html
    \-> metadata0.json
    \-> question1.html
    \-> metadata1.json
```

## **3.4 Database Folder Parser**

This module focuses on parsing the database folder created by the previous module and storing it in the respective data structures in the memory. This module works in almost a similar manner to the previous module, but in a reverse fashion, so instead of writing the files, we read the files and store the data back into the respective data structures. This module is called whenever we need to generate a quiz using Autoquiz.

### 3.5 Quiz Generator

The quiz generator is the module that focuses on the inputs and outputs of the new quiz. This module generates the new quiz in memory after tagging, filtering, and selecting output based on difficulty.

The input required for this module is a markdown file that can be read by the MD2QTI module. This means that it should have a @quiz title and @quiz description tag. The input which must be a .md file must also contain a “generate token” with the following tag:

```
@generate {"tags": list, "count": int, "difficulty": int}
```

The module also supports a list of multiple queries for the @generate arguments.

```
@generate [{"tags": list, "count": int, "difficulty": int}, {"tags": list, "count": int, "difficulty": int}]
```

#### Example:

##### *Single query*

```
@generate [{"tags": ["bit","os"], "count": 5, "difficulty": 0.5}]
```

##### *Multiple queries*

```
@generate [
  {
    "tags": [
      "bit",
      "os"
    ],
    "count": 5,
    "difficulty": 0.5
  },
  {
    "tags": [
      "network"
    ],
    "count": 5,
    "difficulty": 0.3
  }
]
```

This “generate tag” states the following:

1. Generate 5 questions with tags “bit” and “os”. These questions should be of medium to hard difficulty such that only 50% of the students answered them correctly in the previous quizzes.
2. Generate 5 questions with the tag “network”. These questions should be of hard to very hard difficulty such that only 30% of the students have answered them correctly in the previous quizzes.

The Autoquiz generator will then remove the generate tag from the file and replace it with ten questions such that these can be read by MD2QTI.

### *3.5.1 Tagging and filtering*

Tagging is one of the most important parts of processing the query. Before we can filter questions based on tags, we need to tag all the questions such that we can extract the essence of each of the question’s descriptions. To do so, we use the NLTK library.

#### *NLTK*

NLTK is a popular python library for working with language data. It comes with a set of text processing libraries with help with classification, tokenization, stemming, tagging, parsing, and semantic reasoning [17]. We can do the following with the NLTK library [18].

*Example of tokenization using NLTK:*

At eight o'clock on Thursday morning ... Arthur didn't feel very good.

['At', 'eight', "o'clock", 'on', 'Thursday', 'morning', 'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']

*Example of Part-of-speech tagging using NLTK:*

At eight o'clock on Thursday morning ... Arthur didn't feel very good.

[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN'), ('Arthur', 'NNP'), ('did', 'VBD'), ("n't", 'RB'), ('feel', 'VB'), ('very', 'RB'), ('good', 'JJ'), ('.', '.')] ]

Based on this, the tags for the questions are selected based on the following:

1. POS tagged Nouns
2. Quiz Name
3. Course Name

Selecting the quiz name and course along with the nouns ensures that all the required keywords that can be used to define the essence of the question can be captured.

This approach is not perfect and does seem to leave out the broad topic if it's not part of the quiz title or course name. A better way to handle this would be by using machine learning which has been discussed in the future scope section of this report.

Once all the questions in the corpus are tagged, we can filter the questions based on the tags of the query. Once this is done, we select the question k closest questions to the difficulty specified by the user in the query.

### 3.5.2 Selecting questions based on difficulty

Once the questions are filtered based on the tags, the filtered list is sorted based on the difficulty, and the top k closest questions to the difficulty scores are chosen using a variant of binary search.

For example, the figure below shows all the questions with given difficulties in ascending order. In case the user specifies that they want questions of difficulty level - 0.8, our algorithm will select the following questions with the given difficulty level.

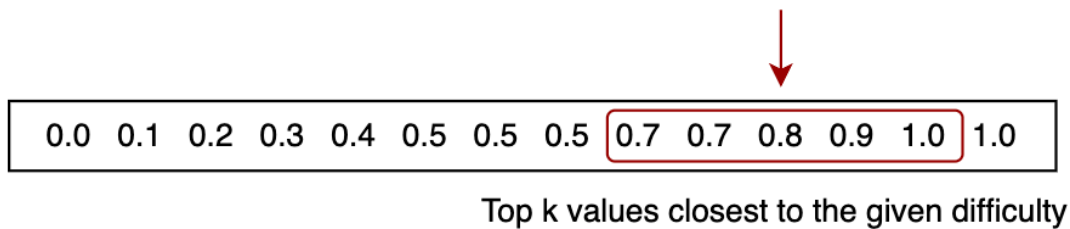


Fig 20: Top k chosen values for a given difficult

### 3.5.3 Converting HTML to MD

All the question description is stored in HTML format. This is the default format supported by Canvas. However, MD2QTI required all the questions to be in markdown format for them to be processed. Therefore, once all the questions are selected, we need to convert them into markdown format. This is done by using the “markdownify” python module which makes it very simple to convert HTML into markdown.

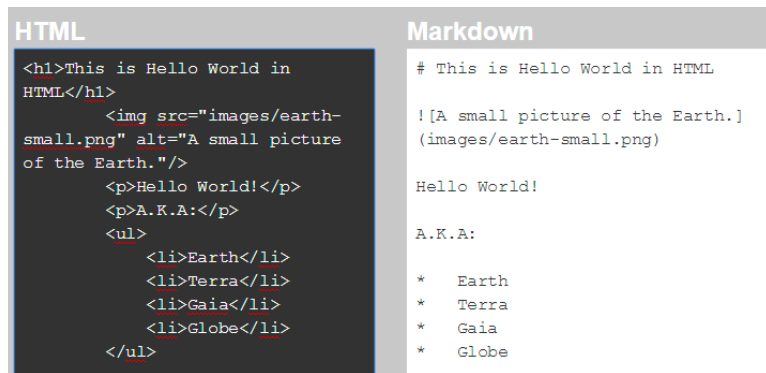


Fig 21: converting HTML to markdown.

### 3.6 Zip Creator

The quiz uploader essentially uses the python subprocess module to call the MD2QTI program which is included in the Autoquiz package. The module uses the output file that was generated by the Quiz generator module and provides its path to the Md2QTI program.

```
python MD2QTI.py quiz.md
```

In case this command fails, the submodule throws an error that is produced by the MD2QTI.py program and quits. If the command is successful, the module will produce a zip file that can then be uploaded to Canvas using the method as stated in the Background and History Section, Under Canvas.

## 4. USING AUTOQUIZ

Autoquiz needs to be set up as a python repository before using the tool. To do so, The repository can be cloned from <https://github.com/architjain123/autoquiz>. A Python 3 virtual environment needs to be set up and activated for the local repository. The required libraries can be installed using the command below:

```
pip install -r requirements.txt
```

Autoquiz supports two commands:

1. Download - to download the courses and create a database.
2. Generate - to autogenerate the quiz and the QTI zip package.

Information about the commands can be accessed using the following command:

```
python autoquiz.py --help
```

### 4.1 Download command

This command will download courses and create a database folder in the working directory.

```
python autoquiz.py download [course_name]
```

It downloads courses based on the regex in the optional “course\_name” argument. In case no “course\_name” argument is provided, it will download all the courses in the database.

### 4.2 Generate command

This command will generate a QTI zip file in the current working directory.

```
python autoquiz.py generate {file_path} {database_path}
```

It creates a quiz by modifying the file at the “file\_path” by using the questions found in the “database\_path”. It will also create an ExportedQTI.zip file that can be imported into Canvas.

[] - optional    {} - required



## 5. SETTING DEVELOPMENT ENVIRONMENT

Before the development of the Canvas Autoquiz tool could begin, an alternative canvas environment was required to be set up. This is because courses in Canvas can only be created by administrators which also have many other privileges such as adding users, instructors, and other administrators. Given that SJSU uses a hosted version of canvas that is used by all the departments, we were unable to get administrator access. We will call this version the production version.

Moreover, having our instance of Canvas LMS allows for better debugging and control. For these reasons, it was decided to set up our instance of Canvas for the development of the Canvas Autoquiz tool, we will call this instance the development instance. The tool was extensively tested on the development instance before it was used on the production version.

The development version of Canvas was set up based on the instructions found on the Canvas Github page. It is important to note that this setup was extremely difficult as the instructions on the page were outdated. The difficulties faced are explained in the Difficulties and Challenges section of the report. The machine used was hosted on AWS EC2 Ohio availability zone and was of the following specification:

- t2 large
- 2 vCPUs
- 8 GiB Memory
- 150 GB gp2 SSD

The machine was assigned an Elastic IP address as it was only active during the development of the tool due to the high rental cost. The development server was added to the authors' personal domain.

## 6. DIFFICULTIES AND CHALLENGES

There were many challenges faced during the project. However, the most difficult challenge to overcome was setting up our instance of Canvas. This was because Canvas is a complex project requiring a very powerful machine and can be deployed in multiple ways.

### 6.1 Problem with setting up Canvas

The documentation provided by Instructure to set up our version of canvas for testing was incomplete and kept on leading to failures. This made it very difficult for the author to start working on canvas as the installation kept on failing.

Eventually, the problems with the installation were identified and rectified. Please note that even though we faced these issues, others might face different issues as the documentation of canvas installation continues to evolve.

The major problem faced was with the way ruby was being installed in the documentation. Although it is stated in the official documentation to install ruby as a local user, it was identified that we need to install ruby globally. Moreover, there were issues with the installation of docker, docker-compose, and yarn when following the official guide. We have specified the code to rectify the issues we face in the subsections below. These can be fixed by running the following commands before starting the quick installing guide found on GitHub.

#### 6.1.1 Ruby setup using RVM

```
sudo apt-add-repository -y ppa:rael-gc/rvm
sudo apt-get update
sudo apt-get install rvm
sudo usermod -a -G rvm $USER
```

(Exit and login)

```
rvm user gemsets
CFLAGS="-Wno-error=implicit-function-declaration" rvm install 2.6.6
```

### 6.1.2 Installing Docker and Docker Compose

```
sudo apt-get update
sudo apt-get install \
  apt-transport-https \
  ca-certificates \
  curl \
  gnupg \
  lsb-release
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

```
echo \
  "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-
keyring.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/
docker.list > /dev/null
```

```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
sudo usermod -aG docker ${USER}
```

(Exit and login)

```
sudo curl -L "https://github.com/docker/compose/releases/download/
1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/
docker-compose
```

```
sudo chmod +x /usr/local/bin/docker-compose
```

### 6.1.3 Installing YARN

```
curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/
apt/sources.list.d/yarn.list
sudo apt-get update && sudo apt-get install yarn=1.19.1-1
```

## 7. FUTURE WORK AND IMPROVEMENTS

### 7.1 Question Tagging

Currently, the question tagging is done based on the question description and the respective responses to the question. Although this might capture some of the keywords required for effective tagging, it still leaves a lot of room for improvement in cases where the broad sense of the topic cannot be captured by the description of the problem.

For example - a question such as “Convert 100101011 from binary to hexadecimal” falls under the category of bit manipulation; however, the current tagger cannot capture this detail. We need another tool that can provide this information.

The easiest way to do so would be to use Machine learning to train a classifier that identifies question types based on the question description. There are already many projects that tag the programming language of a particular question. We can use similar models and train them on our models to produce results of our liking.

### 7.2 Web Application

Another possible improvement that would increase the usability of Autoquiz would be to create a web interface. This would be a central application where all users using different instances of canvas can access their specific instance of canvas by entering their token and Canvas API endpoint. Having a user interface would make it very easy for the instructors to view the quiz before uploading it to canvas. Having a user interface would also allow instructors to swap questions on the fly before generating the final version of the quiz.md file to be uploaded to Canvas.

## 8. CONCLUSION

Canvas Autoquiz reduces the effort of instructors by auto-generating quizzes. The tool fetches historical quiz data from Canvas. The tool then calculates the difficulty of the questions, merges similar questions and their statistics, and tags the questions based on question text. It creates a file system-based database to store these questions. Instructors can use this database to auto-generate quizzes based on the tags and difficulty. These auto-generated quizzes can then be imported into Canvas.

The database generated by instructors can be shared to allow easy transfer of quiz data. Autoquiz can also be used to import Canvas quizzes to any other online-learning-based software that uses the QTI standard.

**REFERENCES**

- [1] Canvas, “What is canvas?,” What is Canvas? - Instructure Community, 16-Mar-2022. [Online]. Available: <https://community.canvaslms.com/t5/Canvas-Basics-Guide/What-is-Canvas/ta-p/45>. [Accessed: 24-Apr-2022].
- [2] C. Etherington, “Why colleges and universities are adopting canvas,” eLearningInside, 26-Jun-2019. [Online]. Available: <https://news.elearninginside.com/why-colleges-and-universities-are-adopting-canvas/>. [Accessed: 24-Apr-2022].
- [3] P. Hill., “North American higher Ed LMS market share by enrollments: A consolidating market,” eliteRate, 03-Oct-2018. [Online]. Available: <https://eliterate.us/na-he-lms-market-share-enrollments-for-2012-2018/>. [Accessed: 24-Apr-2022].
- [4] P. Hill, “State of higher ed LMS market for US and Canada: Year-end 2020 edition,” PhilOnEdTech, 04-Feb-2021. [Online]. Available: <https://philonedtech.com/state-of-higher-ed-lms-market-for-us-and-canada-year-end-2020-edition/>. [Accessed: 24-Apr-2022].
- [5] University of Rhode Island, “Basics of the Unix Philosophy,” Chapter 1: Philosophy. [Online]. Available: [https://homepage.cs.uri.edu/~thenry/resources/unix\\_art/ch01s06.html](https://homepage.cs.uri.edu/~thenry/resources/unix_art/ch01s06.html). [Accessed: 24-Apr-2022].
- [6] M. Polakowski, “Canvas LMS - Pros & Cons,” Selleo, 20-Apr-2020. [Online]. Available: <https://selleo.com/blog/canvas-lms-pros-and-cons>. [Accessed: 25-Apr-2022].
- [7] Valencia College, “Canvas Dashboard: Canvas Essentials Sandbox,” Canvas Dashboard. [Online]. Available: <https://online.valenciacollege.edu/courses/26030/pages/canvas-dashboard>. [Accessed: 24-Apr-2022].
- [8] UBC, “Get started with the Canvas API,” The University of British Columbia Learning Analytics. [Online]. Available: <https://learninganalytics.ubc.ca/for-students/canvas-api/>. [Accessed: 24-Apr-2022].
- [9] Instructure Community, “How do I manage API access tokens as an admin?,” Instructure Community, 16-Apr-2022. [Online]. Available: <https://community.canvaslms.com/t5/Admin-Guide/How-do-I-manage-API-access-tokens-as-an-admin/ta-p/89>. [Accessed: 24-Apr-2022].
- [10] Instructure Community, “How do I import quizzes from QTI packages?,” Instructure Community, 16-Apr-2022. [Online]. Available: <https://community.canvaslms.com/t5/Instructor-Guide/How-do-I-import-quizzes-from-QTI-packages/ta-p/1046>. [Accessed: 24-Apr-2022].
- [11] Markdown Guide, “Getting started: An overview of Markdown,” Markdown Guide. [Online]. Available: <https://www.markdownguide.org/getting-started/>. [Accessed: 24-Apr-2022].

- [12] Bit Blog Editorial Team, “What is Markdown & How It Can Help You Write Faster,” Bit Blog, 03-Jun-2021. [Online]. Available: <https://blog.bit.ai/what-is-markdown/>. [Accessed: 24-Apr-2022].
- [13] Instructure Community, “What is QTI,” Instructure Community, 25-Mar-2020. [Online]. Available: <https://community.canvaslms.com/t5/Canvas-Question-Forum/What-is-QTI/m-p/201389>. [Accessed: 24-Apr-2022].
- [14] IMS Global Learning Consortium, “IMS Question & Test Interoperability Specification Overview,” IMS Global Learning Consortium. [Online]. Available: <https://www.imsglobal.org/question/index.html>. [Accessed: 24-Apr-2022].
- [15] E. Nam, “Understanding the levenshtein distance equation for beginners,” Medium, 27-Feb-2019. [Online]. Available: <https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0>. [Accessed: 24-Apr-2022].
- [16] N. Raut, “What is Hamming Distance?,” Tutorials Point, 31-Dec-2018. [Online]. Available: <https://www.tutorialspoint.com/what-is-hamming-distance>. [Accessed: 24-Apr-2022].
- [17] E. Loper and S. Bird, “5. Categorizing and Tagging Words,” in NLTK: The natural language toolkit, S.l.: s.n.
- [18] E. Loper and S. Bird, “7. Extracting Information from Text,” in NLTK: The natural language toolkit, S.l.: s.n.
- [19] S. Kim, “SJSU-CS-Systems-Group/MD2QTI,” GitHub. [Online]. Available: <https://github.com/SJSU-CS-systems-group/MD2QTI>. [Accessed: 24-Apr-2022].
- [20] G. Poore, “Gpoore/text2qti: Create quizzes in QTI format for canvas from Markdown-based plain text,” GitHub. [Online]. Available: <https://github.com/gpoore/text2qti>. [Accessed: 24-Apr-2022].
- [21] A. Gupta, “Online Quizzes: Uses and Benefits,” Aurosolar. [Online]. Available: <https://aurosolar.com/blog/Online-Quizzes-Uses-and-Benefits>. [Accessed: 24-Apr-2022].
- [22] University of Central Florida, “UCFOPEN/CANVASAPI: Python API wrapper for Instructure's canvas LMS,” GitHub. [Online]. Available: <https://github.com/ucfopen/canvasapi>. [Accessed: 24-Apr-2022].
- [23] A. Haapala, “Ztane/Python-Levenshtein,” GitHub. [Online]. Available: <https://github.com/ztane/python-Levenshtein>. [Accessed: 24-Apr-2022].