Paper ID #37127

Work In Progress: Beyond Textbook: An Open Educational Resource Platform that Generates Course-Specific E-Textbooks

Barney Wei

Mingyu Zheng

Mohammadreza Karamsoltani

RUI ZENG

Hamid S. Timorabadi

Hamid Timorabadi received his BSc, MASc, and PhD degrees in Electrical Engineering from the University of Toronto. He has worked as a project, design, and test engineer as well as a consultant to industry. His research interests include the application of digital signal processing in energy systems and computer networks. He also has deep interest in engineering education and the use of technology to advance the learning experience of undergraduate students.

Work In Progress: Beyond Textbook: An Open Educational Resource Platform that Generates Course-Specific E-Textbooks

Abstract

Beyond Textbook (BT) is an Open Educational Resources (OER) platform that combines an e-textbook generation algorithm with a website interface. The frontend allows users to upload lecture notes. Then, given a list of topics, the backend matches each topic with the most relevant lecture notes, merges these lecture notes into one file, and finally generates a customized e-textbook for users to view and download. BT is developed to provide instructors and students with an e-textbook that is customized to specific requirements in a course. BT offers a zero-cost avenue to deliver and access curriculum content in a standardized, but collaborative and dynamic manner. The goal is to reduce the financial barrier to education, allow students to have access to up-to-date educational content, and leverage modern technology to improve pedagogy and learning. We proceeded with a trial run of BT involving both instructors and students in a first-year course and collected their feedback. Survey results identified that all participants found BT to be a useful educational tool and would use it upon its release.

1.0 Introduction and Motivation

Open Educational Resources (OER) are educational resources freely available to everyone. OER links to a global educational movement that began about 20 years ago [1] to replace the existing platform of paid textbooks with the goals of allowing students to have access to up-to-date and relevant-to-course content, leveraging modern technology to improve teaching and learning, and reducing the financial barrier to education [2]. OER addresses three main problems stemming from paid textbooks: lack of variety in content delivery in terms of both perspective and medium, out-of-date information, and textbook affordability. Textbook publishers try to address out-of-date information by creating new editions, nevertheless, this forces students to purchase a new revision of the textbook obsoleting the older ones. This exacerbates the problem of textbook affordability which is a problem that is only getting worse as the textbook market is dominated by five publishers that control over 80% of the course material [3]. Moreover, students can no longer save money by sharing, buying used or renting hard copies of textbooks in the current transition from hard copy e-textbooks. Studies show that students who use OER also perform academically similar to students who use paid textbooks [4].

One of the main barriers to OER adoption is the difficulty to find content targeted and custom-designed for individual courses [5]. As a result, we wish to introduce an OER platform, Beyond Textbook (BT), that is tailored toward individual courses. We envision BT to contain an amalgamation of key features of existing OER platforms: Coursera, where we will partner with institutions to deliver course-specific content led by professional instructors with certificates of completion; Khan Academy, where we will provide an informal learning option that allows users to search and learn specific concepts; YouTube, where we will allow video contributions from anyone to distill concepts in a straightforward and easily-understandable manner ranked by user votes and comments; Wikipedia, where we will allow typed contributions from anyone which will then undergo a peer-review process to deliver quality and accurate content; Chegg, where we will provide detailed solutions to questions and offer discounted tutoring services; Piazza, where we will include a community question-and-answer platform led by professional instructors. Unique to BT, it will provide course-specific interactive and user-contributable e-textbooks that will undergo peer-review to enable quality content. However, to start, we will be scoping down the project and only focusing on generating e-textbooks from lecture notes to allow course-specific content.

2.0 Existing Work

One of the state-of-the-art solutions for e-textbook generation is BBookX [6] which is an automated web-based recommender system composed of two independent components. The first is a table of contents generator that links book chapters to entire Wikipedia articles. The second is an interactive user interface that allows manual real-time collaborative textbook creation by users. The problem with the first is that it becomes the responsibility of the users to locate their topics of interest in full Wikipedia articles, which are also written towards a specific target audience and may be difficult to understand. The problem with the second is that it cannot scale easily to replace other textbooks due to the need for large portions of manual input. In 2005, Philip Parker filed a patent [7] to allow the automatic generation of content which he used to author more than 200,000 books [8]. However, the algorithm is genre-specific and produces very generic content with a target audience of people who are not proficient in searching the Internet. Thus, to the best of our knowledge, no existing solution can generate consistent and coherent text, convey relevant, concise and easy-to-understand information to explain course-specific topics and at the same time, easily scale to replace textbooks from other courses.

3.0 Beyond Textbook

BT is an online work-in-progress OER platform that delivers customized, credible, and continuously updated e-textbooks that closely follow course content at no cost in a standardized, but collaborative and dynamic manner. BT allows users to upload lecture notes through its website interface and then given a list of topics, users will be able to auto-generate a personalized e-textbook from the uploaded notes. The backend algorithm determines the relatedness metrics used to distill the database of notes using the provided topics to create the e-textbook. Our solution is unique as it incorporates user inputs both for determining the topics as well as the contents of a customized course textbook. Having users, students and instructors, involved in the process of determining what would be included in the course textbook make BT a solution that is unique and reliable.

3.1 Backend Matching Algorithm

The backend matching algorithm that takes typed lecture notes and user topic queries as inputs and eventually outputs the customized e-textbook document is implemented using three key concepts: (1) a keyword extraction algorithm proposed by Studart Rose et al. (2010) [9], (2) a term frequency-inverse document frequency (TF-IDF) numerical statistic [10], and (3) GloVe vectors [11].

The first stage of the backend involves the pre-processing of typed notes provided by instructors and students. Supported file formats consist of DOC, DOCX, ODT, and TXT. These documents will be converted into the open-source ODT file format using the LibreOffice command-line interface. Conversion to ODT file type is chosen to ensure standardized file formatting while being open-sourced and license-free to improve accessibility.

The second stage involves the parsing of ODT documents which consists of two separate tasks. The first task is keyword extraction while the second is metadata collection. In the first task, the content of each ODT document is categorized into three buckets: (1) file name, (2) headings, and (3) paragraph content. The keyword extraction algorithm proposed by Rose et al. (2010) [9] is then run on the content of each bucket. For each bucket, the extracted keywords and their computed TF-IDF score (ratio of their number of occurrences against the number of words) in the document are collected. In the second task, useful metadata such as the creation user ID, file creation timestamp, file name, file path, file size, number of characters, number of pages, number of words, etc. of each ODT document is collected. The data collected from both tasks are stored in a MongoDB database. While the data collected from the first task is essential in the later stages for query-to-document matching, data collected from the second task may be used as filters to ignore certain documents.

In the third stage, the system takes a list of user queries of topics they want the generated e-textbook to contain as input. This list is ordered in the sense that the topics covered by the content of the generated e-textbook will follow the order of this list. The same keyword extraction algorithm proposed by Rose et al. (2010) [9] is run on each query. To allow better robustness in the matching process in the next stage, GloVe vectors [11] are used to identify the ten most related synonyms in decreasing order of relatedness to each extracted keyword. At this point, each query has been decomposed into one or more keywords with each keyword having ten other synonyms most associated with it.

Continuing to the fourth stage, we use the TF-IDF numerical statistic to match each user query to the most relevant ODT file based on whichever file has the highest overall TF-IDF score. We do this by first assigning weights to the three buckets: (1) file name, (2) headings, and (3) paragraph content in decreasing value with the file name having the highest relevancy. We also assign different weights to the type of match: (1) keyword match or (2) synonym match with the keyword match having the highest weight, and thus, highest relevance. A keyword match refers to finding an exact match of a keyword extracted from the user query with a keyword extracted from an ODT document. A synonym match refers to finding an exact match of one of the synonyms of a keyword extracted from the user query with a keyword extracted from an ODT document. Now, given a user query, we first try to conduct a keyword match for each of its extracted keywords by querying the MongoDB database to obtain their TF-IDF score for each bucket of each ODT document. If this is not successful, we fall back and try to conduct a synonym match. If this continues to be unsuccessful, we skip this extracted keyword in question and thus, ignore its contribution to the relatedness scoring. All these TF-IDF scores are multiplied by their associated weights depending on bucket type and type of match. These scores are then summed up for each ODT document to obtain its overall TF-IDF score associated with the specific user query in question. The ODT document that has the highest overall TF-IDF score then becomes the most relevant document to the user query. We continue this technique for each user query and ultimately, each user query will be associated with the most relevant ODT document.

The first and second stages constitute the ingestion process of the typed notes provided by instructors and students as input. The third and fourth stages constitute the query-to-document matching process where each user query is matched with the most relevant ODT document. Finally, the fifth stage merges the relevant ODT documents in order of the user queries to generate the customized e-textbook. If identical ODT documents exist in the list to be merged, subsequent identical documents are ignored. Figure 1 illustrates all these backend stages.

3.2 Frontend Website Service

For users to access our online platform, they first need to register or log in with a unique username and a 6-character-length password. After logging in, they can browse through all uploaded materials by categories and keywords. The hierarchy we use to guide users includes University, Course, and Topic (e.g., “arrays”). These options can be searched and selected from dropdown menus if they already exist. Otherwise, users can easily add a new instance. Appendix D provides screenshots and details of the Website Service Interface.

To contribute new materials to the platform, users must first navigate to the page of a specific topic of a specific university’s course. Then, they need to add some additional information about the uploaded note. For instance, they can indicate whether this note is a lecture, tutorial, or practical session note, by selecting the category from the dropdown menu. Another example is that they can indicate the year and semester of this note, like 2021 Fall. Finally, after adding all the necessary information, contributors can upload their notes. A submission response page will show up and guide contributors to submit another query or return to the home page.

On the other hand, users who want to build customizable textbooks from uploaded resources must create a syllabus first, either by typing in a list of topics on their own or uploading a PDF syllabus to let our algorithm do the extraction work. The syllabus extraction algorithm will generate a list of topics that users can modify before they proceed to the next step. The next step is to add some filters, including semester, instructor, session etc., to the desired textbook. Our backend algorithm will fetch all the materials with the highest relevance to users’ needs and generate an e-textbook. Then, users will be brought to the preview page from where they can also go back if they find any modifications needed. Finally, if they are satisfied with the current version, they can click the download button to save a local copy of the generated e-textbook. Figure 1 illustrates the system block diagram that includes both the frontend and backend.

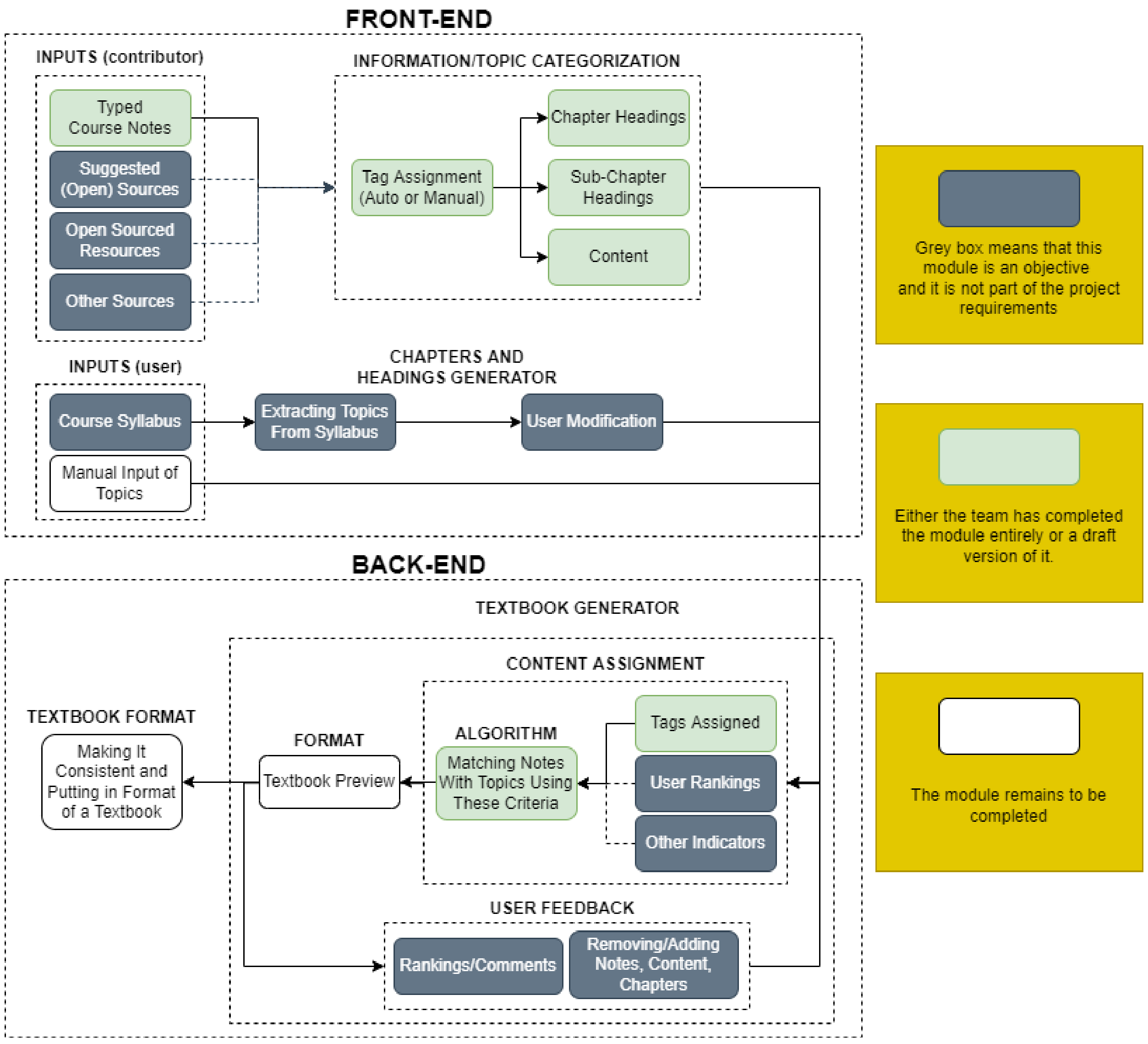

FRONT-END
INPUTS (contributor)
Typed Course Notes
Suggested (Open) Sources
Open Sourced Resources
Other Sources
INFORMATION/TOPIC CATEGORIZATION
Tag Assignment (Auto or Manual)
Chapter Headings
Sub-Chapter Headings
Content
INPUTS (user)
Course Syllabus
Manual Input of Topics
CHAPTERS AND HEADINGS GENERATOR
Extracting Topics From Syllabus
User Modification
BACK-END
TEXTBOOK GENERATOR
CONTENT ASSIGNMENT
ALGORITHM
Matching Notes With Topics Using These Criteria
Tags Assigned
User Rankings
Other Indicators
FORMAT
Textbook Preview
TEXTBOOK FORMAT
Making It Consistent and Putting in Format of a Textbook
USER FEEDBACK
Rankings/Comments
Removing/Adding Notes, Content, Chapters
Grey box means that this module is an objective and it is not part of the project requirements
Either the team has completed the module entirely or a draft version of it.
The module remains to be completed

Figure 1: System Block Diagram of BT.

4.0 Evaluation and Student Reception

To evaluate the usability and effectiveness of BT and receive real feedback from its intended users, we decided to have a class of first-year computer engineering students use BT for their first programming course. For the evaluation process, students were given regular course notes and textbook recommendations as their main learning materials and used BT as a parallel source of information to better facilitate their learning process. After making BT available to students, we surveyed them to receive feedback on the usability, effectiveness, and accuracy of BT. The main conclusions and takeaways from the surveys received from the students are listed below.

Fifteen first-year computer engineering students enrolled in their first programming course participated in our survey. They were all questioned on what learning materials they use for their course and they all mentioned that they do not use the recommended textbook for the course and instead use professors' notes or other sources. Also, the majority of them (80%) indicated that they generally find textbooks to be only somewhat useful or not useful at all which indicates the lack of usability and resourcefulness of textbooks in general among post-secondary students. On the other hand, over 85% of the participants found BT to be either very useful or useful and the remaining 15% found it to be somewhat useful. In addition, all of the participants mentioned that they would use BT as a new educational platform/tool upon its full release. We also collected a set of recommendations and improvements that can be performed to enhance BT's functionality and usability. Figure 2 compares the usefulness of various course materials based on survey results:

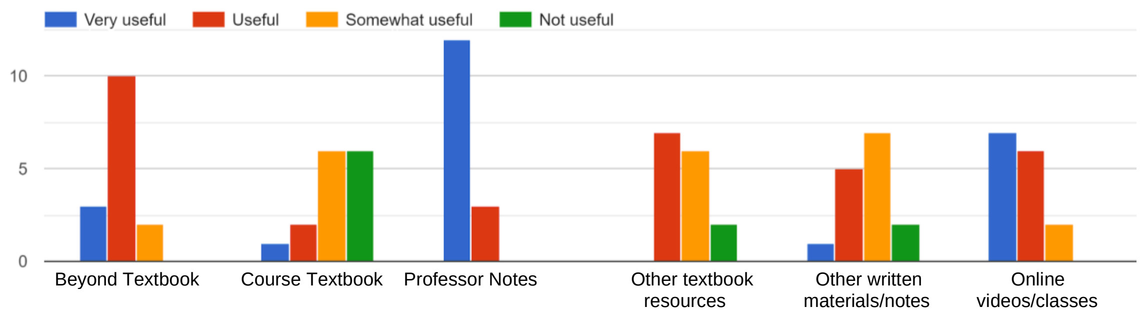


Figure 2: Usefulness of various course materials from the student survey.

5.0 Conclusion and Future Work

BT has been under development since Fall 2021. By sharing course-related resources, this platform aims to enable post-secondary instructors and students to deliver and access up-to-date course-related content at zero cost in a standardized, collaborative and dynamic manner. The platform is hosted on a website where users can upload lecture notes, search for academic resources, and build customized textbooks. Meanwhile, our backend algorithms will ensure that the generated textbook is most suitable to the user's needs and facilitate the generating process.

In order to collect feedback from the end-users of our product, we surveyed fifteen first-year computer engineering students at our institution who were enrolled in their first coding course. Among all participants, over 85% of them found BT to be either very useful or useful and the remaining 15% found it to be somewhat useful, considerably higher than course textbooks that only rated 20% very useful or useful. Also, all participants mentioned that they would use BT as a new educational platform/tool upon its complete release. Students also recommended a few improvements and features to be implemented for the complete release of BT.

In the future, BT will include improvements to complement its core features such as having different user types and privileges, copyright checking, a note vetting process, and an automated process for ensuring notation consistency across all uploaded notes. Furthermore, BT will include many add-on features, such as a question-and-answer platform. Unlike traditional textbooks, the content of the e-textbooks generated by BT will include videos, user comments (Q&A), and other interactive elements to enhance education and understanding. The nature of e-textbooks also allows quick searching for keywords and related content. Moreover, a mobile app version of the BT, in addition to the web version, will be released facilitating accessibility.

References

[1] Bliss, T J and Smith, M. 2017. A Brief History of Open Educational Resources. In: Jhangiani, R S and Biswas-Diener, R. (eds.) *Open: The Philosophy and Practices that are Revolutionizing Education and Science.* Pp. 9–27. London: Ubiquity Press. DOI: https://doi.org/10.5334/bbc.b. License: CC-BY 4.0

[2] "Open Education," *SPARC*. [Online]. Available: https://sparcopen.org/open-education/. [Accessed: 19-Oct-2021].

[3] Cailyn Nagle, Kaitlyn Vitez, and U.S. PIRG Education Fund, "FIXING THE BROKEN TEXTBOOK MARKET Second Edition." Public Interest Research Group, Jun-2020.

[4] J. Hilton, "Open educational resources and college textbook choices: a review of research on efficacy and perceptions," *Springer Link*, 19-Feb-2016. [Online]. Available: https://link.springer.com/article/10.1007/s11423-016-9434-9. [Accessed: Jun-2021].

[5] D. Munro, J. Omassi, and B. Yano, "Step One: What Are OER, Why Are They Important, and What are the Barriers to Adoption?," OER Student Toolkit, 26-May-2016. [Online]. Available: https://opentextbc.ca/studenttoolkit/chapter/step-one-what-are-oer/. [Accessed: 19-Oct-2021].

[6] B. K. Pursel, C. Liang, S. Wang, Z. Wu, K. Williams, B. Brautigam, S. Saul, H. Williams, K. Bowen, and L. C. Giles, "BBookX: An Automatic Book Creation Framework," *ResearchGate*, Apr-2016. [Online]. Available: https://www.researchgate.net/publication/303518885_BBookX_Design_of_an_Automated_Web-based_Recommender_System_for_the_Creation_of_Open_Learning_Content. [Accessed: Jul-2021].

[7] P. M. Parker, "Method and apparatus for automated authoring and marketing."

[8] N. Cohen, "He wrote 200,000 books (BUT computers did some of the work)," *The New York Times*, 14-Apr-2008. [Online]. Available: https://www.nytimes.com/2008/04/14/business/media/14link.html. [Accessed: July-2021].

[9] Rose, Stuart & Engel, Dave & Cramer, Nick & Cowley, Wendy. (2010). Automatic Keyword Extraction from Individual Documents. 10.1002/9780470689646.ch1.

[10] T. Mei, "Demystify TF-IDF in Indexing and Ranking," *Medium*, 22-Dec-2019. [Online]. Available: https://ted-mei.medium.com/demystify-tf-idf-in-indexing-and-ranking-5c3ae88c3fa0. [Accessed: 22-Nov-2021].

[11] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014.

Appendix A - Project Requirements

To aid in defining the scope of BT in its initial form, Table A.1 illustrates its project requirements.

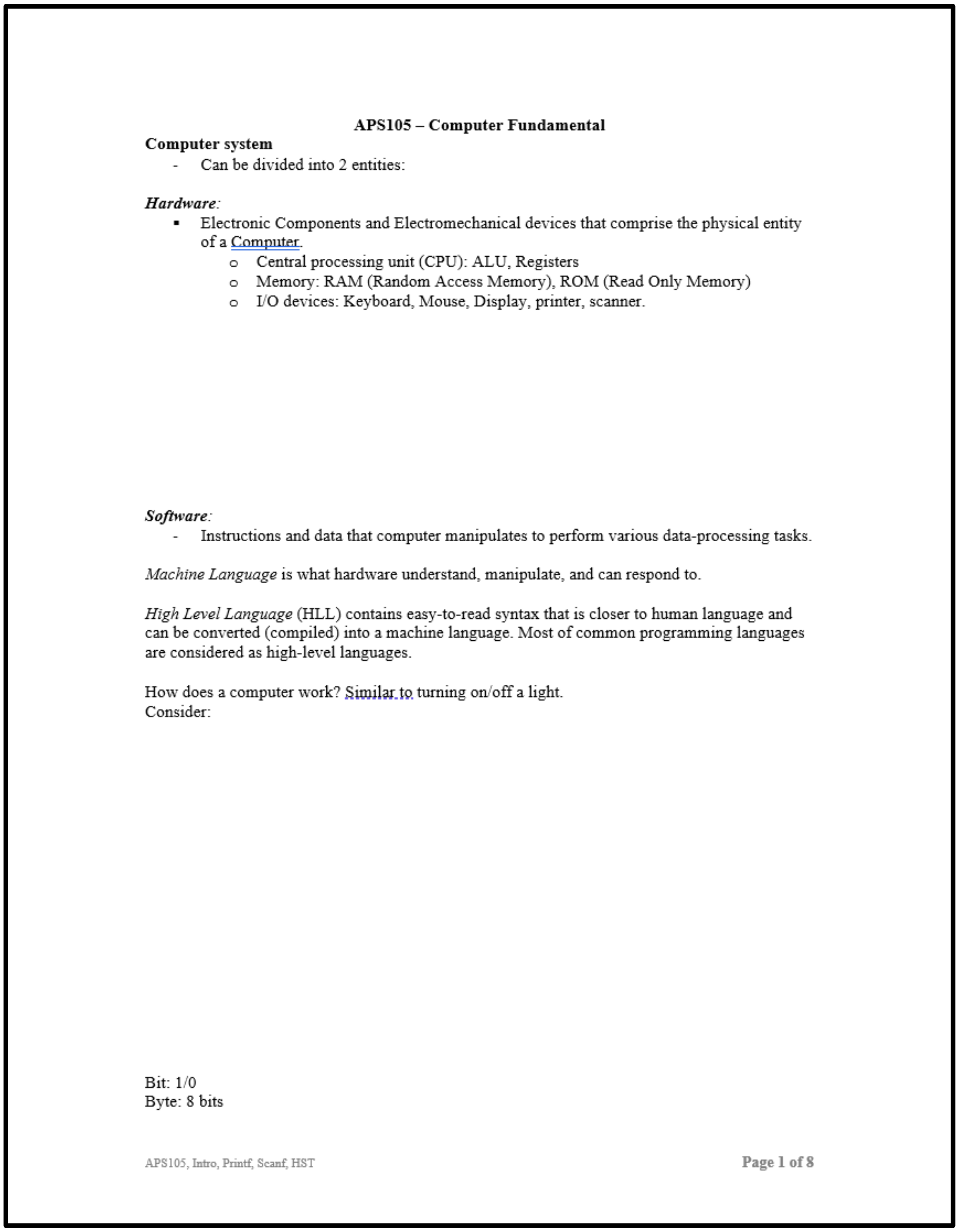
Table A.1: Requirements Table

Category	ID	Project Requirement	Description
Functional Requirement	F1	Store PDF files along with additional information uploaded by contributors.	Contributors include but are not limited to students and professors.
	F2	Implement an algorithm that returns files corresponding to given topics.	Given a list of topics, the backend shall return a list of files corresponding to the topics.
	F3	The application shall be able to combine the list of files together and create a new PDF file.	Combine multiple PDF files together into one new PDF file.
	F4	Implement an algorithm that can look into files and identify topics in them.	For example, if a user uploads a note file of a lecture that covers 'loops' and 'if condition', the algorithm shall identify these two topics and return a topic list.
	F5	Return combined textbook reaches 6/10 satisfactory score on content relatedness from 25 students	Digital survey for university students to provide scores of their satisfaction based on relatedness of the generated textbook.
Constraint	C1	Compatibility: Microsoft Edge	The web interface must be compatible with the Microsoft Edge browser.
Objective	O1	Minimize generation time	The generation time includes time in F2 and F3. We aim to optimize our algorithm to minimize query time.
	O2	Implement a feedback system in the application.	We wish to have a page for users to provide feedback and rate the PDF files in the database. The rating can be useful when we try to implement F2 and achieve O3.
	O3	Maximize content relatedness (Return top-rated notes	In F2, we return a file that corresponds to a topic. If there are multiple files in the database that match the topic, we aim to

		combination)	return the one with the best rating. This rating can be provided by the feedback system in O2. Since users' ratings can reflect content-relatedness, choosing the best rating will contribute to O3.
	O4	Maximize compatibility	We aim to expand the compatibility to include more types of browsers, like Google Chrome, Safari, Firefox etc.
	O5	Implement an algorithm that can look into files and break down files according to topics	For example, if a user uploads a note file of a lecture 3 that covers two topics: 'loops' and 'if condition', the algorithm shall be able to separate the file into two files that only have one topic in each.
	O6	Implement an algorithm that can rate files automatically by given criteria.	The application can rate files without the feedback system in O2.

Appendix B - Sample Typed Notes as Input

Figures B.1 to B.8 are screenshots of the typed notes from the first lecture of APS 105 (Computer Fundamentals) course that allowed to concretize inputs to BT.



APS105 – Computer Fundamental

Computer system

- Can be divided into 2 entities:

Hardware:

- Electronic Components and Electromechanical devices that comprise the physical entity of a Computer.
    - Central processing unit (CPU): ALU, Registers
    - Memory: RAM (Random Access Memory), ROM (Read Only Memory)
    - I/O devices: Keyboard, Mouse, Display, printer, scanner.

Software:

- Instructions and data that computer manipulates to perform various data-processing tasks.

Machine Language is what hardware understand, manipulate, and can respond to.

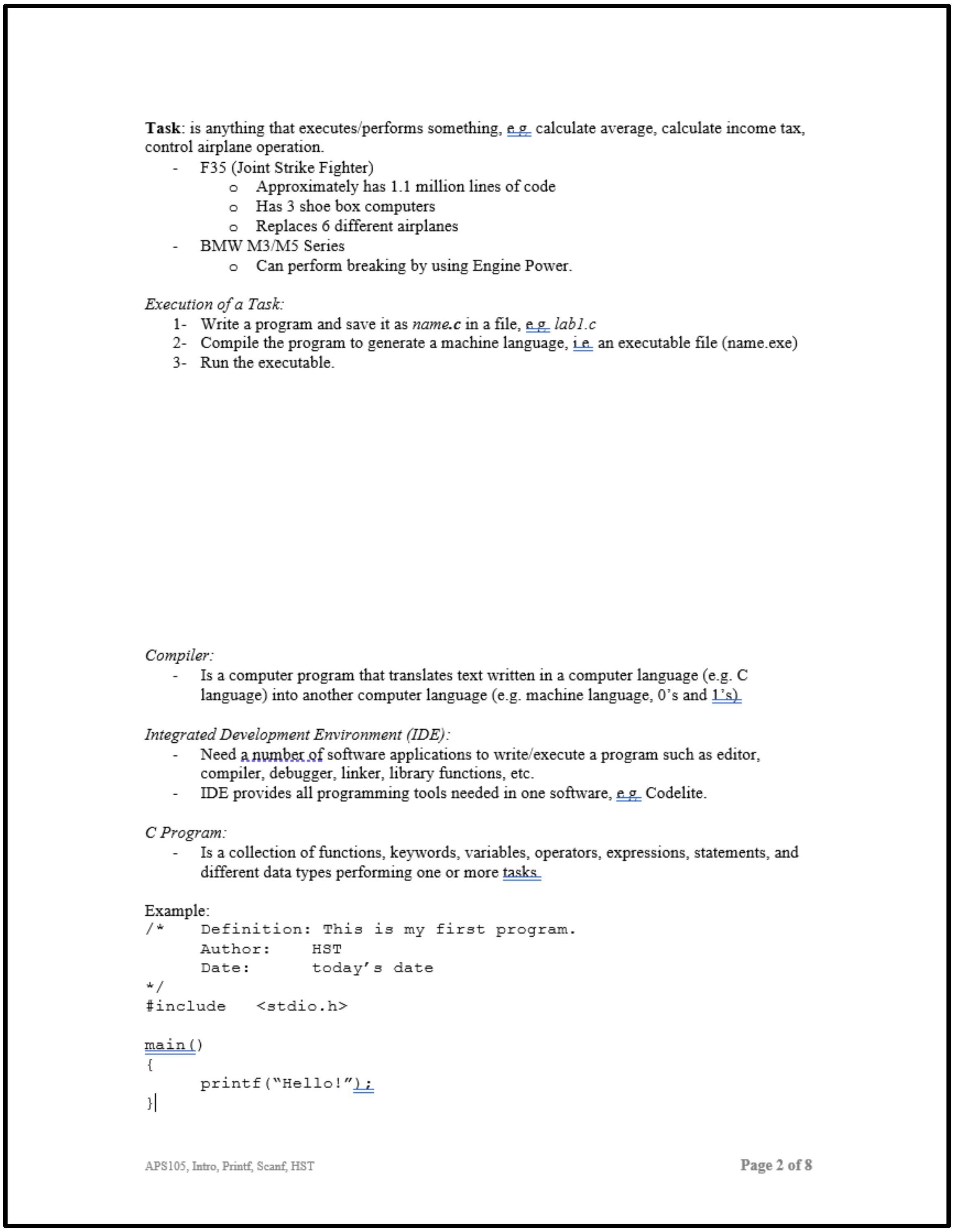
High Level Language (HLL) contains easy-to-read syntax that is closer to human language and can be converted (compiled) into a machine language. Most of common programming languages are considered as high-level languages.

How does a computer work? Similar to turning on/off a light.
Consider:

Bit: 1/0
Byte: 8 bits

APS105, Intro, Printf, Scanf, HST **Page 1 of 8**

Figure B.1: Screenshot of page One of APS 105 (Computer Fundamentals) Course notes, Lecture 1 – an Introduction to *Printf()* and *Scanf()*.



Task: is anything that executes/performs something, e.g. calculate average, calculate income tax, control airplane operation.

- F35 (Joint Strike Fighter)
  - Approximately has 1.1 million lines of code
  - Has 3 shoe box computers
  - Replaces 6 different airplanes
- BMW M3/M5 Series
  - Can perform breaking by using Engine Power.

Execution of a Task:

1- Write a program and save it as *name.c* in a file, e.g. *lab1.c*
2- Compile the program to generate a machine language, i.e. an executable file (name.exe)
3- Run the executable.

Compiler:

- Is a computer program that translates text written in a computer language (e.g. C language) into another computer language (e.g. machine language, 0's and 1's)

Integrated Development Environment (IDE):

- Need a number of software applications to write/execute a program such as editor, compiler, debugger, linker, library functions, etc.
- IDE provides all programming tools needed in one software, e.g. Codelite.

C Program:

- Is a collection of functions, keywords, variables, operators, expressions, statements, and different data types performing one or more tasks.

Example:

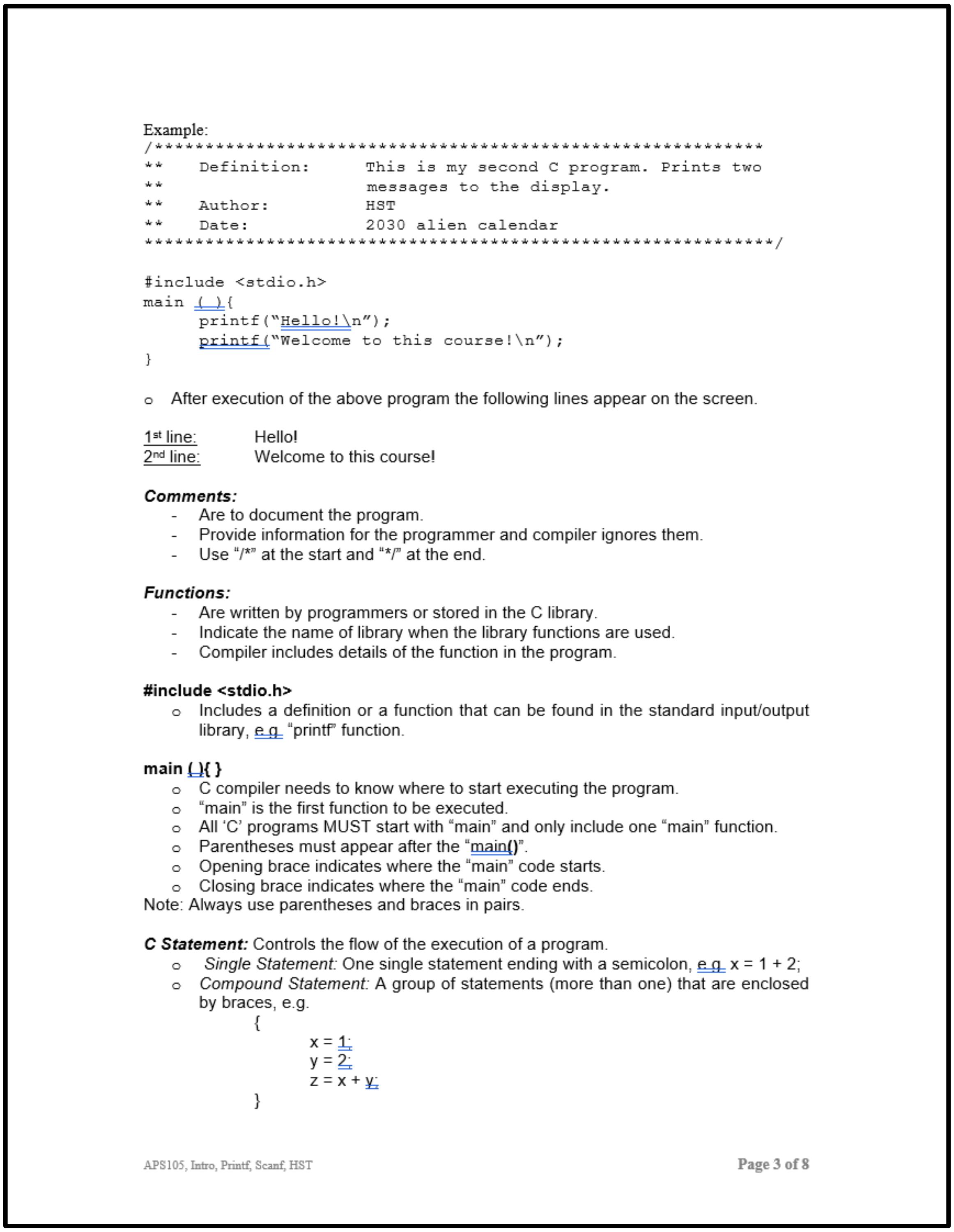
```
/*    Definition: This is my first program.
      Author:    HST
      Date:      today's date
*/
#include   <stdio.h>

main()
{
      printf("Hello!");
}
```

APS105, Intro, Printf, Scanf, HST

Page 2 of 8

Figure B.2: Screenshot of page Two of APS 105 (Computer Fundamentals) Course notes, Lecture 1 – an Introduction to *Printf()* and *Scanf()*.



Example:

```
/*************************************************************
**    Definition:     This is my second C program. Prints two
**                    messages to the display.
**    Author:         HST
**    Date:           2030 alien calendar
**************************************************************/

#include <stdio.h>
main ( ){
      printf("Hello!\n");
      printf("Welcome to this course!\n");
}
```

- After execution of the above program the following lines appear on the screen.

1st line:	Hello!
2nd line:	Welcome to this course!

Comments:

- Are to document the program.
- Provide information for the programmer and compiler ignores them.
- Use "/*" at the start and "*/" at the end.

Functions:

- Are written by programmers or stored in the C library.
- Indicate the name of library when the library functions are used.
- Compiler includes details of the function in the program.

#include <stdio.h>

- Includes a definition or a function that can be found in the standard input/output library, e.g. "printf" function.

main ( ){ }

- C compiler needs to know where to start executing the program.
- "main" is the first function to be executed.
- All 'C' programs MUST start with "main" and only include one "main" function.
- Parentheses must appear after the "main**()**".
- Opening brace indicates where the "main" code starts.
- Closing brace indicates where the "main" code ends.

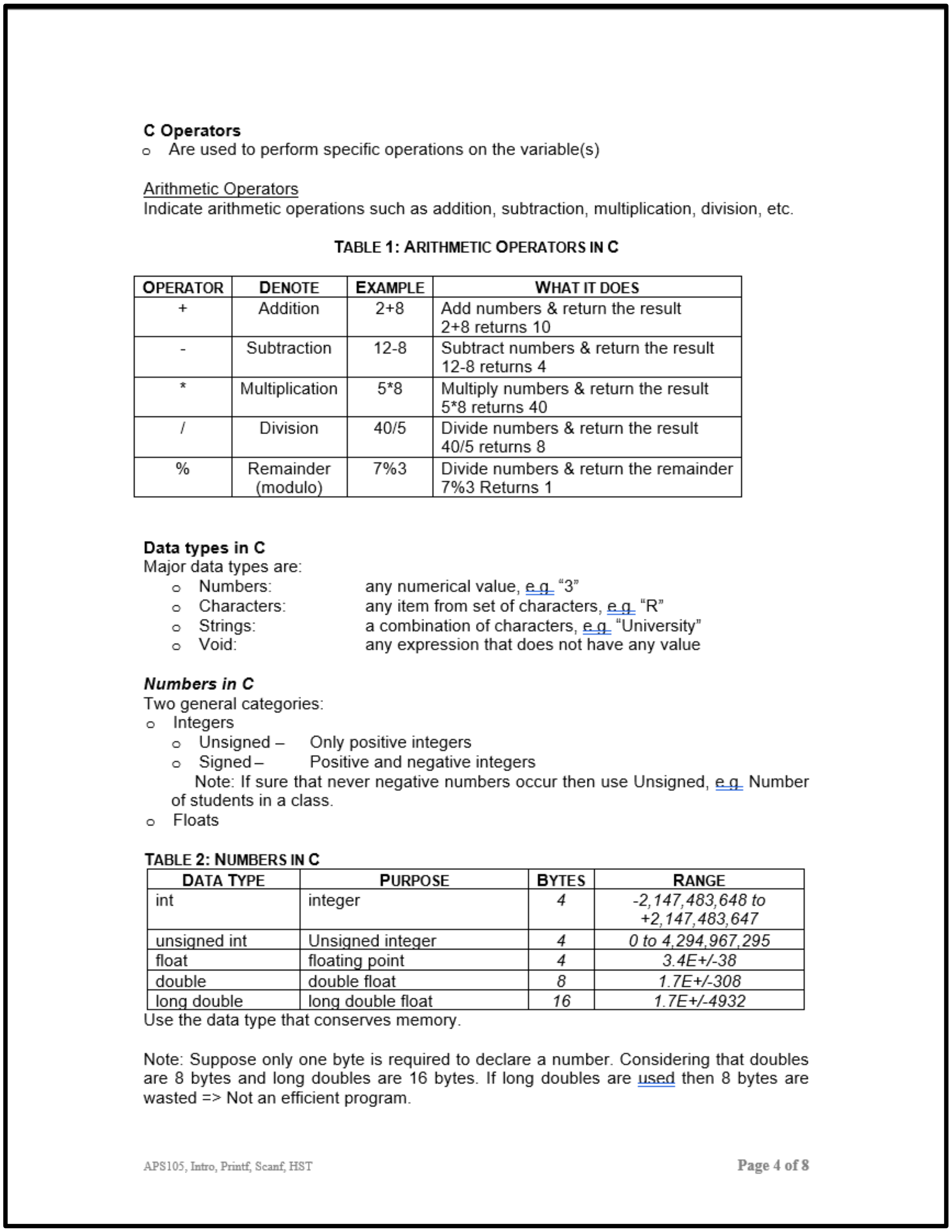
Note: Always use parentheses and braces in pairs.

C Statement: Controls the flow of the execution of a program.

- *Single Statement:* One single statement ending with a semicolon, e.g. x = 1 + 2;
- *Compound Statement:* A group of statements (more than one) that are enclosed by braces, e.g.

```
{
        x = 1;
        y = 2;
        z = x + y;
}
```

Figure B.3: Screenshot of page Three of APS 105 (Computer Fundamentals) Course notes, Lecture 1 – an Introduction to *Printf()* and *Scanf()*.



C Operators

- Are used to perform specific operations on the variable(s)

Arithmetic Operators

Indicate arithmetic operations such as addition, subtraction, multiplication, division, etc.

TABLE 1: ARITHMETIC OPERATORS IN C

OPERATOR	DENOTE	EXAMPLE	WHAT IT DOES
+	Addition	2+8	Add numbers & return the result 2+8 returns 10
-	Subtraction	12-8	Subtract numbers & return the result 12-8 returns 4
*	Multiplication	5*8	Multiply numbers & return the result 5*8 returns 40
/	Division	40/5	Divide numbers & return the result 40/5 returns 8
%	Remainder (modulo)	7%3	Divide numbers & return the remainder 7%3 Returns 1

Data types in C

Major data types are:

Numbers:	any numerical value, e.g. “3”
Characters:	any item from set of characters, e.g. “R”
Strings:	a combination of characters, e.g. “University”
Void:	any expression that does not have any value

Numbers in C

Two general categories:

- Integers
  - Unsigned – Only positive integers
  - Signed – Positive and negative integers

  Note: If sure that never negative numbers occur then use Unsigned, e.g. Number of students in a class.
- Floats

TABLE 2: NUMBERS IN C

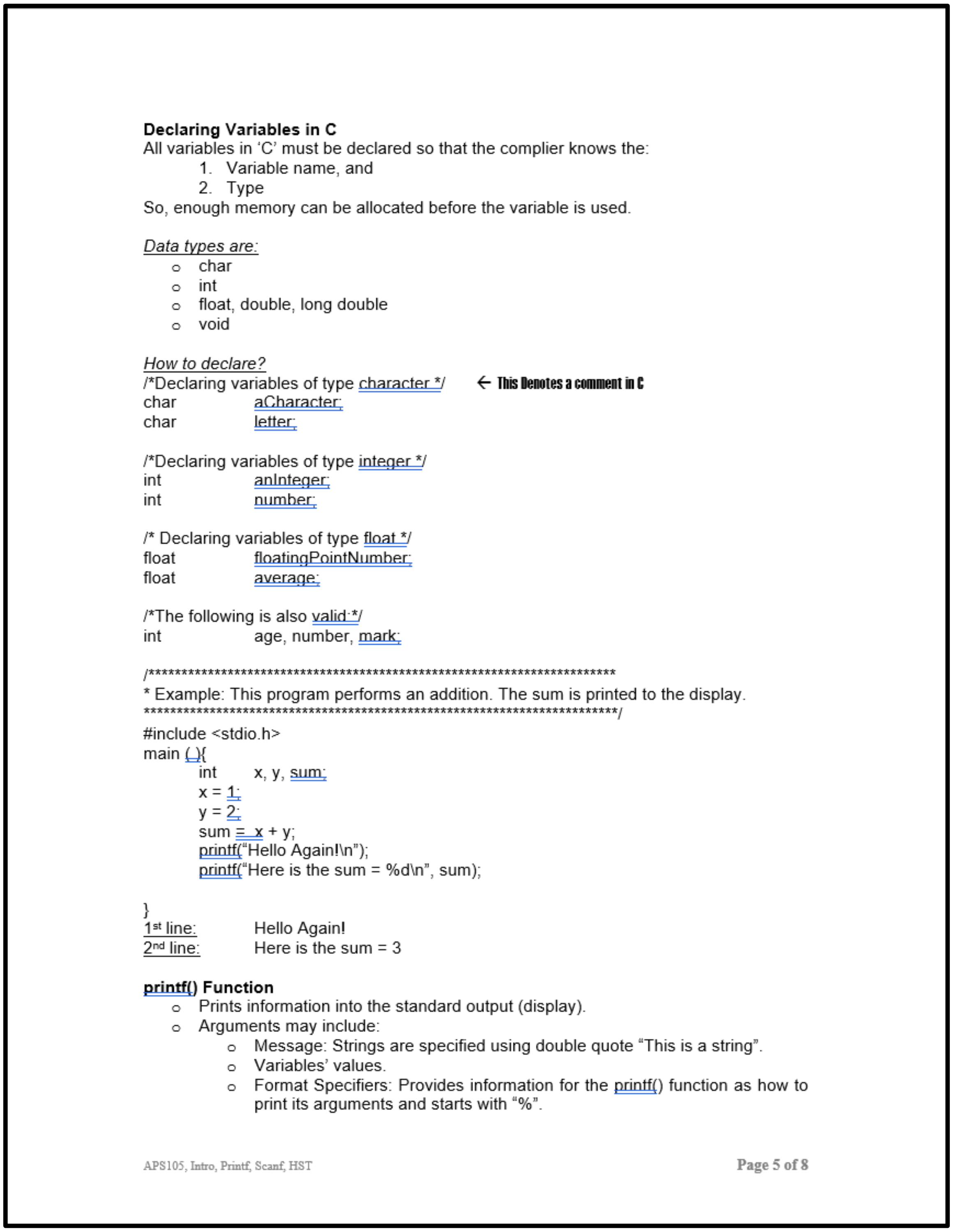
DATA TYPE	PURPOSE	BYTES	RANGE
int	integer	*4*	*-2,147,483,648 to +2,147,483,647*
unsigned int	Unsigned integer	*4*	*0 to 4,294,967,295*
float	floating point	*4*	*3.4E+/-38*
double	double float	*8*	*1.7E+/-308*
long double	long double float	*16*	*1.7E+/-4932*

Use the data type that conserves memory.

Note: Suppose only one byte is required to declare a number. Considering that doubles are 8 bytes and long doubles are 16 bytes. If long doubles are used then 8 bytes are wasted => Not an efficient program.

APS105, Intro, Printf, Scanf, HST   **Page 4 of 8**

Figure B.4: Screenshot of page Four of APS 105 (Computer Fundamentals) Course notes, Lecture 1 – an Introduction to *Printf()* and *Scanf()*.



Declaring Variables in C

All variables in 'C' must be declared so that the complier knows the:

1. Variable name, and
2. Type

So, enough memory can be allocated before the variable is used.

Data types are:

- char
- int
- float, double, long double
- void

How to declare?

```
/*Declaring variables of type character */        ← This Denotes a comment in C
char            aCharacter;
char            letter;

/*Declaring variables of type integer */
int             anInteger;
int             number;

/* Declaring variables of type float */
float           floatingPointNumber;
float           average;

/*The following is also valid */
int             age, number, mark;

/************************************************************************
* Example: This program performs an addition. The sum is printed to the display.
************************************************************************/
#include <stdio.h>
main (){
        int     x, y, sum;
        x = 1;
        y = 2;
        sum = x + y;
        printf("Hello Again!\n");
        printf("Here is the sum = %d\n", sum);

}
```

1st line: Hello Again!
2nd line: Here is the sum = 3

printf() Function

- Prints information into the standard output (display).
- Arguments may include:
  - Message: Strings are specified using double quote "This is a string".
  - Variables' values.
  - Format Specifiers: Provides information for the printf() function as how to print its arguments and starts with "%".

APS105, Intro, Printf, Scanf, HST   **Page 5 of 8**

Figure B.5: Screenshot of page Five of APS 105 (Computer Fundamentals) Course notes, Lecture 1 – an Introduction to *Printf()* and *Scanf()*.

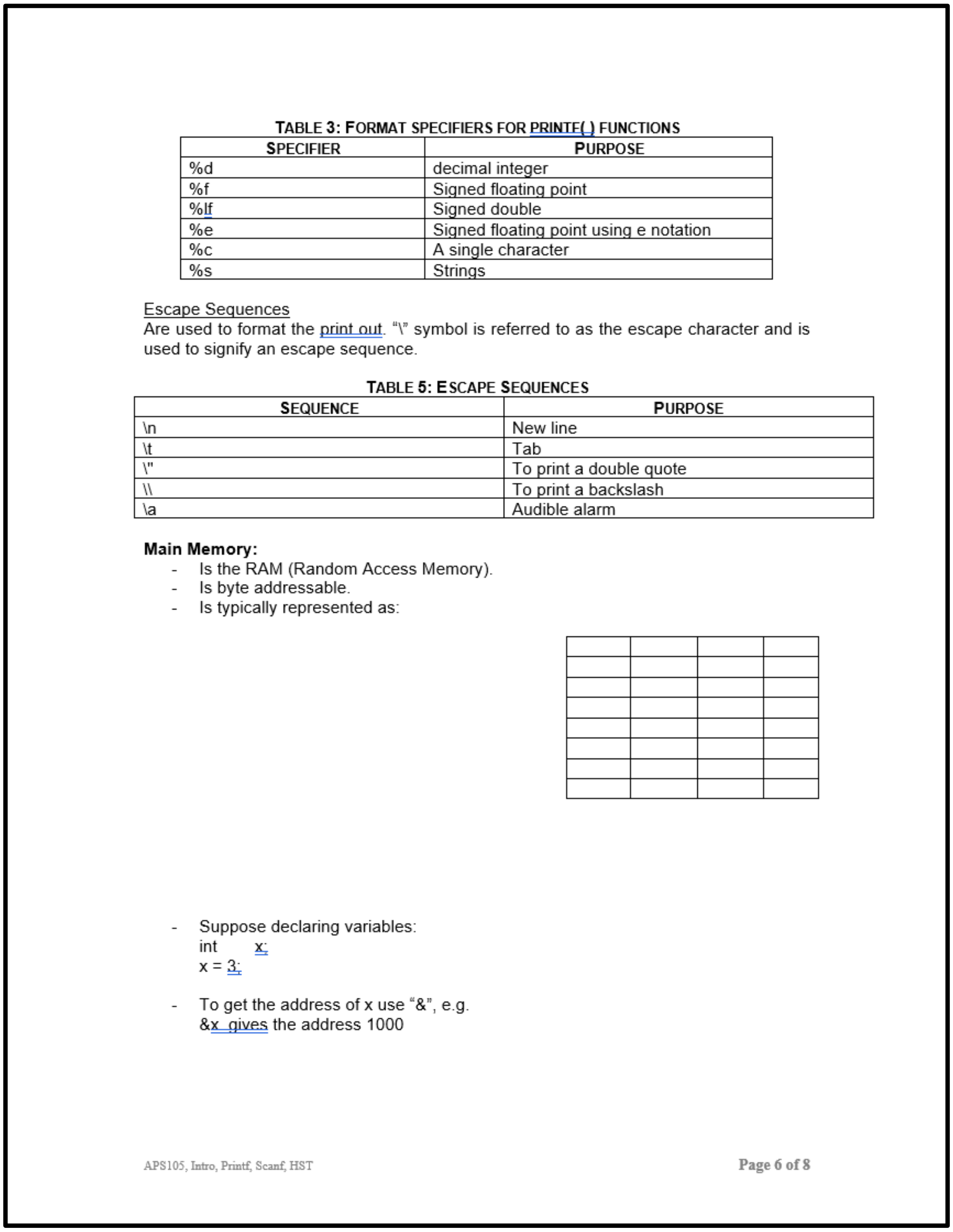


TABLE 3: FORMAT SPECIFIERS FOR PRINTF( ) FUNCTIONS

SPECIFIER	PURPOSE
%d	decimal integer
%f	Signed floating point
%lf	Signed double
%e	Signed floating point using e notation
%c	A single character
%s	Strings

Escape Sequences

Are used to format the print out. "\" symbol is referred to as the escape character and is used to signify an escape sequence.

TABLE 5: ESCAPE SEQUENCES

SEQUENCE	PURPOSE
\n	New line
\t	Tab
\"	To print a double quote
\\	To print a backslash
\a	Audible alarm

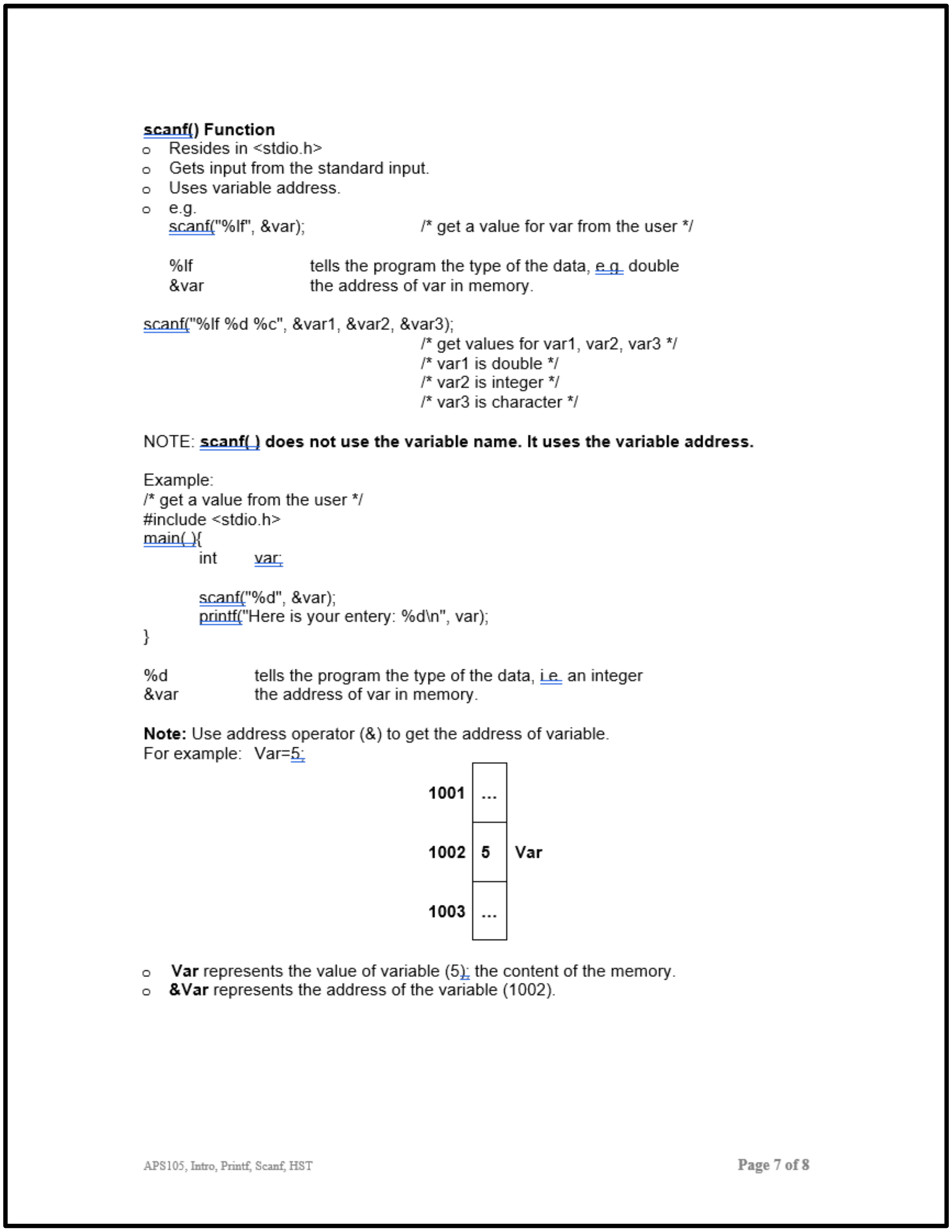
Main Memory:

- Is the RAM (Random Access Memory).
- Is byte addressable.
- Is typically represented as:

- Suppose declaring variables:
  int   x;
  x = 3;

- To get the address of x use "&", e.g.
  &x gives the address 1000

Figure B.6: Screenshot of page Six of APS 105 (Computer Fundamentals) Course notes, Lecture 1 – an Introduction to *Printf()* and *Scanf()*.



scanf() Function

- Resides in <stdio.h>
- Gets input from the standard input.
- Uses variable address.
- e.g.

```
scanf("%lf", &var);            /* get a value for var from the user */

%lf          tells the program the type of the data, e.g. double
&var         the address of var in memory.
```

```
scanf("%lf %d %c", &var1, &var2, &var3);
                               /* get values for var1, var2, var3 */
                               /* var1 is double */
                               /* var2 is integer */
                               /* var3 is character */
```

NOTE: **scanf( ) does not use the variable name. It uses the variable address.**

Example:

```
/* get a value from the user */
#include <stdio.h>
main( ){
        int     var;

        scanf("%d", &var);
        printf("Here is your entery: %d\n", var);
}
```

```
%d          tells the program the type of the data, i.e. an integer
&var        the address of var in memory.
```

Note: Use address operator (&) to get the address of variable.
For example: Var=5;

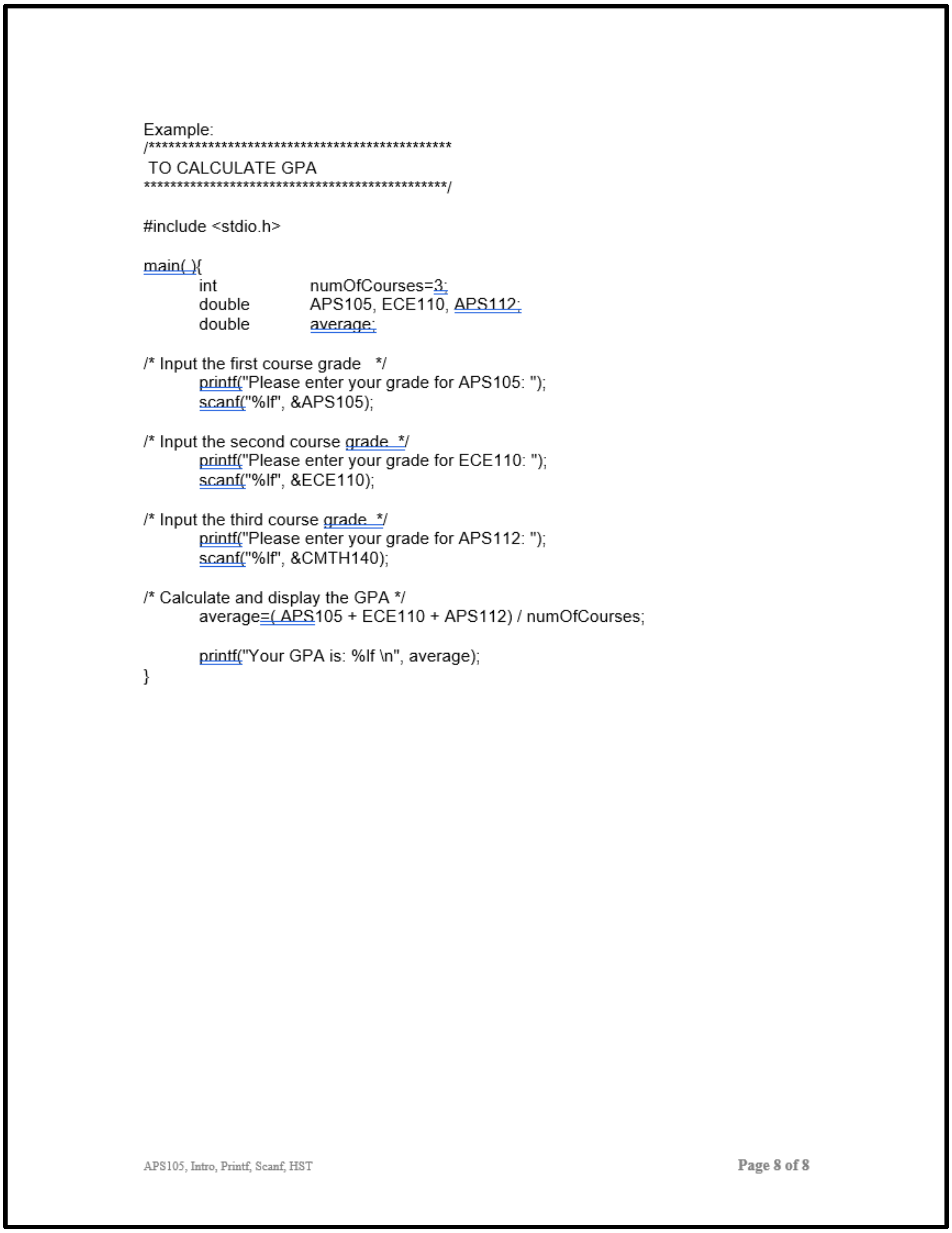
1001	**...**	
1002	**5**	**Var**
1003	**...**	

- **Var** represents the value of variable (5); the content of the memory.
- **&Var** represents the address of the variable (1002).

APS105, Intro, Printf, Scanf, HST

Page 7 of 8

Figure B.7: Screenshot of page Seven of APS 105 (Computer Fundamentals) Course notes, Lecture 1 – an Introduction to *Printf()* and *Scanf()*.



Example:

```
/*********************************************
 TO CALCULATE GPA
*********************************************/

#include <stdio.h>

main( ){
        int             numOfCourses=3;
        double          APS105, ECE110, APS112;
        double          average;

/* Input the first course grade   */
        printf("Please enter your grade for APS105: ");
        scanf("%lf", &APS105);

/* Input the second course grade  */
        printf("Please enter your grade for ECE110: ");
        scanf("%lf", &ECE110);

/* Input the third course grade  */
        printf("Please enter your grade for APS112: ");
        scanf("%lf", &CMTH140);

/* Calculate and display the GPA */
        average=( APS105 + ECE110 + APS112) / numOfCourses;

        printf("Your GPA is: %lf \n", average);
}
```

APS105, Intro, Printf, Scanf, HST

Page 8 of 8

Figure B.8: Screenshot of page Eight of APS 105 (Computer Fundamentals) Course notes, Lecture 1 – an Introduction to *Printf()* and *Scanf()*.

Appendix C - Verification Table

Table C.1 identifies the verification methods to test BT for functionality and correctness.

Table C.1: Verification Table

ID	Project Requirement	Verification Method
1.0	Store PDF files along with additional information uploaded by contributors.	TEST: upload notes and randomly pull 20% from the database to check for existence. Pass if all test files exist, else fail.
2.0	Implement an algorithm that can return a list of files given a topic in decreasing order of topic frequency existing in its content.	TEST: given topics and check if returned files match those topics in decreasing order of topic frequency. Pass if all topics each return a list of files in decreasing order of topic frequency.
3.0	The application shall be able to combine a list of files together and create a new PDF file.	TEST: given a list of files and the ordering for combination, check if the output PDF file contains all the file content without overlapping or missing content. Pass if returned single PDF file for 2 sets of data containing all files in order without overlapping or missing content
4.0	Implement an algorithm that can identify topics from files.	TEST: given a file, and check if returned topics match word(s) that exist in its content. Pass if returned topics match existing word(s) in the file, else fail.
5.0	Compatibility	TEST: directly go through all pages to check if there are any bugs such as broken links. Pass if frontend layout showing content and effect as desired design in Microsoft Edge.
6.0	Return combined textbook output file reaches 6/10 satisfactory score on content relatedness from 25 students	TEST: provide digital surveys to 25 students who have tried out our platform and collect and calculate average satisfaction scores on content relatedness from them. Pass if the average score meets or exceeds 6 out of 10, else fail.

Appendix D - Website Service Interface

Figures D.1 to D.6 are screenshots demonstrating a few samples of the BT interface Website Service.

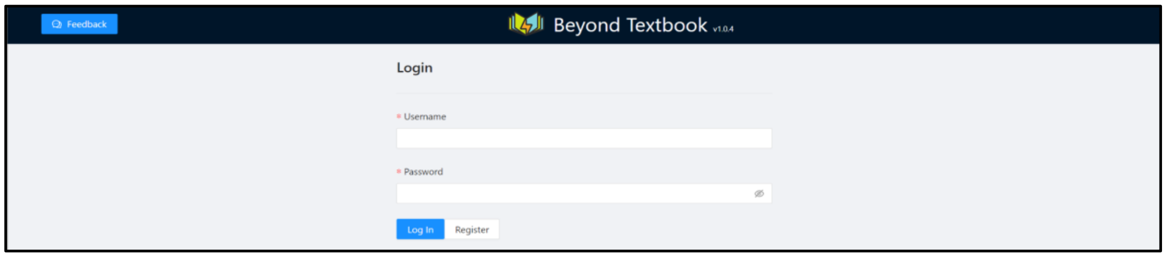


Figure D.1: Shows the login page. The navigation process starts from this login page. This page allows a user to choose to log in or register for a new account. Subsequent navigation on BT's web interface maintains the user's login credential until a logout is processed.

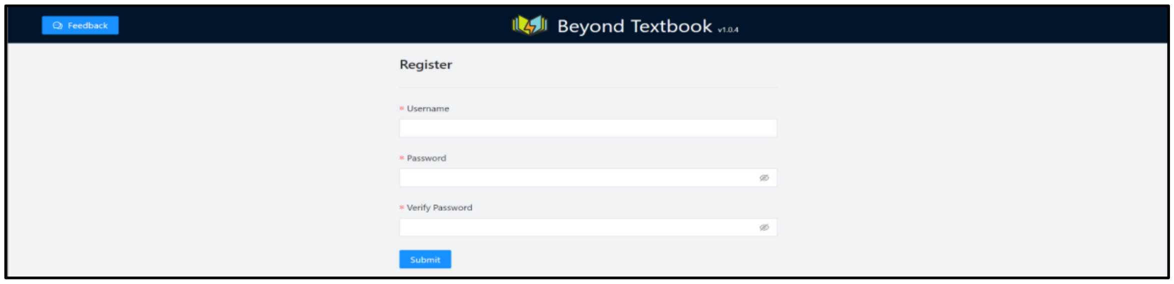
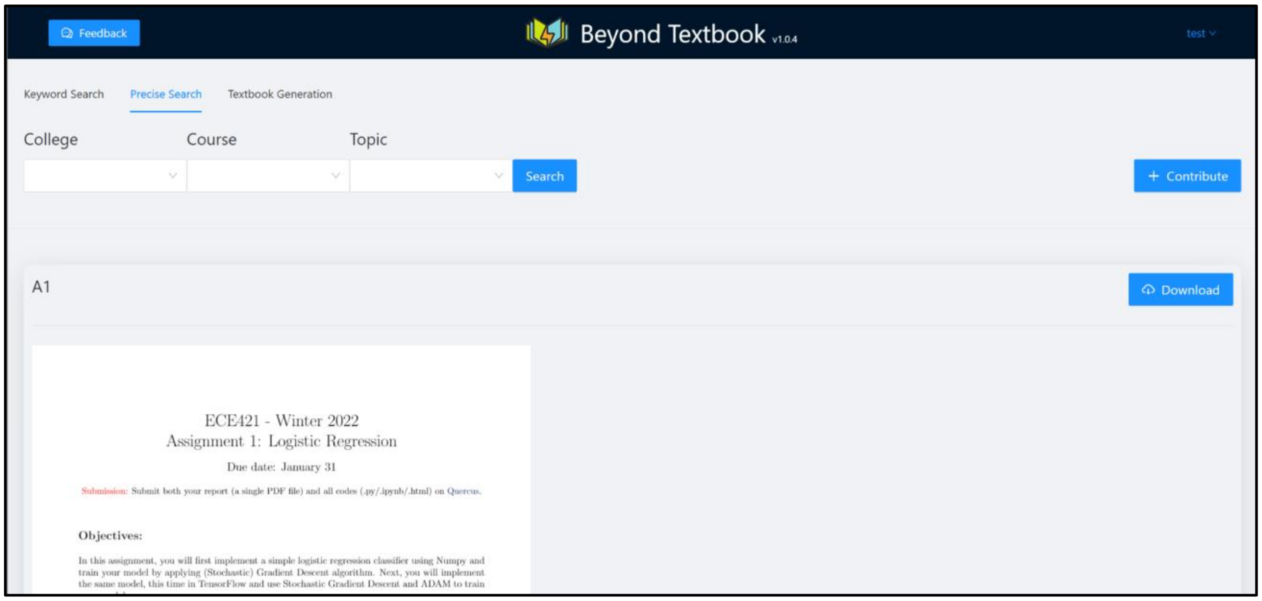


Figure D.2: Shows a registration page that allows a user to sign in to BT and create an account.



ECE421 - Winter 2022
Assignment 1: Logistic Regression

Due date: January 31

Submission: Submit both your report (a single PDF file) and all codes (.py/.ipynb/.html) on Quercus.

Objectives:

In this assignment, you will first implement a simple logistic regression classifier using Numpy and train your model by applying (Stochastic) Gradient Descent algorithm. Next, you will implement the same model, this time in TensorFlow and use Stochastic Gradient Descent and ADAM to train

Figure D.3: Depicts the landing page. The landing page is where a user can choose to search existing notes using keywords (*Keyword Search tab*). This page also allows a search by *college name*, *course name*, or a certain topic name that can be reached via drop down menu.

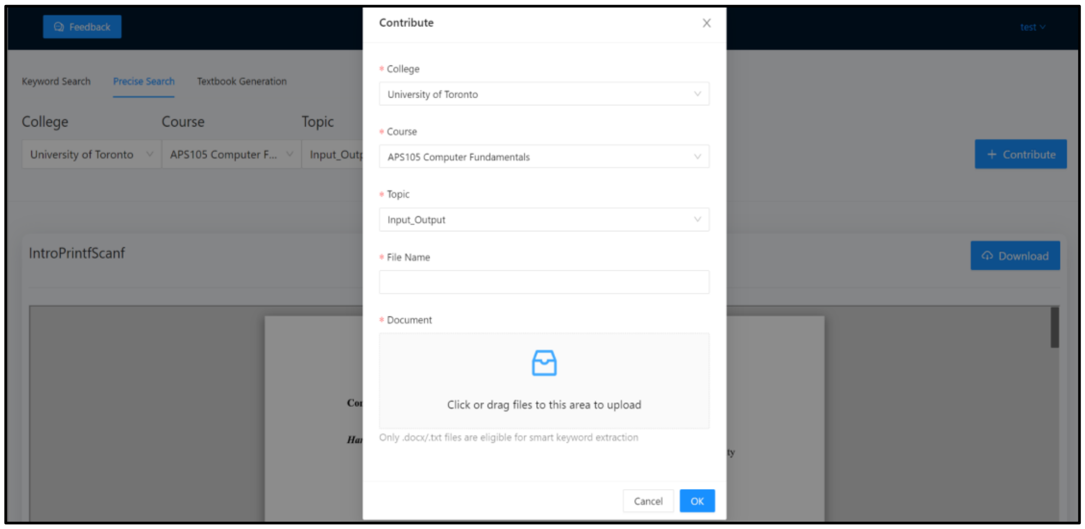
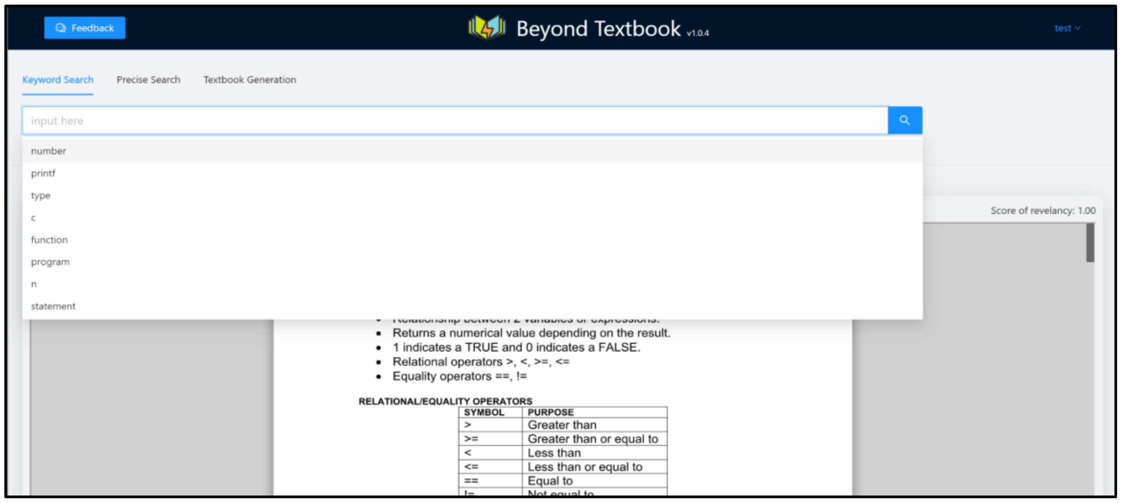


Figure D.4: Illustrates the Contribute window. A user can contribute his/her own notes by clicking on the Contribute button that opens the Contribute window.



- Returns a numerical value depending on the result.
- 1 indicates a TRUE and 0 indicates a FALSE.
- Relational operators >, <, >=, <=
- Equality operators ==, !=

RELATIONAL/EQUALITY OPERATORS

SYMBOL	PURPOSE
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
==	Equal to
!=	Not equal to

Figure D.5: Shows the Keyword Search where user can specify a keyword and hence searching the database content for the keyword. Then any material that contains the keyword will be loaded for the user's access in decreasing order of relevancy.

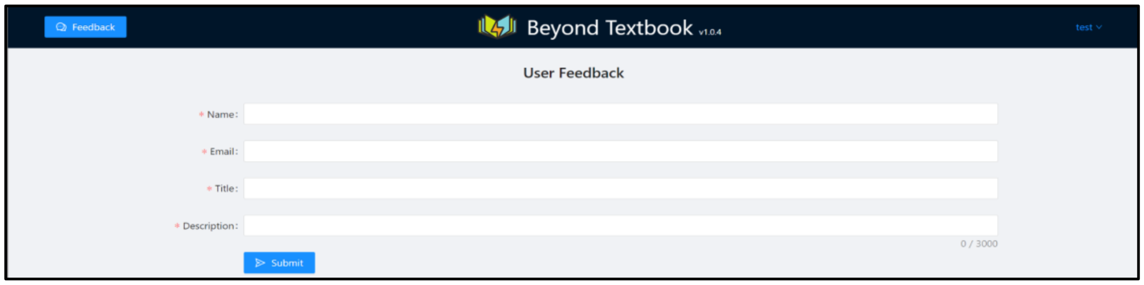


Figure D.6: Depicts the User Feedback page. This page provides users a means of providing feedback, comments, and suggestions that can be considered by staff.