Active Reading Assistant
Programmer's Documentation

Spring Term 2024 - CS 422 Anthony Hornof

Ethan Hyde (EH)
John Hooft Toomey (JHT)
John O'Donnell (JO)
Ben Bushey (BB)

Table of Contents

1. Introduction/Installation Instructions

The following programmer’s documentation covers all of the main parts of our Active Reading Assistant software. It explains how the application’s code, functions, and files work together and covers the use cases of each function within a given module. All code is written in Python using the Visual Studio Code IDE. This document assumes a foundational knowledge of the Python language as well as the MacOS operating system.

The installation requirements to run this system are:

1. tkinter
2. json
3. PyMuPDF
4. Python 3.11.5

These libraries are able to be installed by entering the command ‘pip install {library_name}’ to the terminal. Below is an example of installing the PyMuPDF library on Mac. (See Figure 1.1)

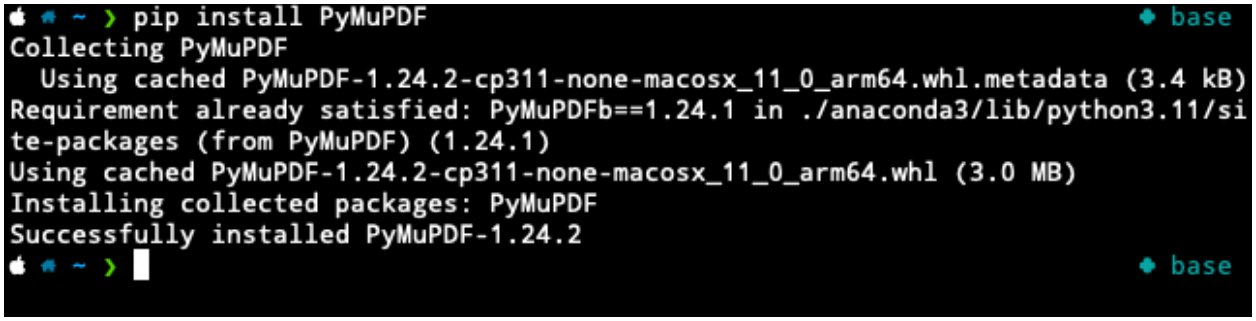

Figure 1.1

2. Active Reading Assistant Source Files

2.1. MongoDB - datastructs.py

This file is responsible for defining and managing the data structures and database interactions for our Active Reading Assistant (ARA) software. It allows the general API functions that grant functionality for our Interactive Learning Hub (ILH), and communicate with our database.

2.1.1 __init__(self, uri, dbname, schema_file) - Database class

This is the initialization function for the Database class. It takes in the parameters self, uri, dbname, and schema_file. The uri is the uri string to connect to the database. The dbname variable is the name of the database used with MongoDB, and the schema_file variable is the JSON file containing our schema format. It creates a MongoClient instance that is used to interact with our server. If it can't connect after 2 seconds, then the server is not running. Otherwise, the server connects to the correct database and assigns it to 'self.db.'

2.1.2 - loadSchema(schema_file)

Used to read and load a JSON schema file. The JSON schema defines the format of the JSON document and is used for validating JSON data.

2.1.3 - getCollection(self, collectionName)

Retrieves a MongoDB collection by name from a connected database. It provided us with a way to access specific collections to be used with later operations.

2.1.4 - deleteSection(self, username, pdfID, targetTitle)

Removes a specific section from a user's notes in the database. This function uses MongoDB's 'update_one' method with an '$pull' operation to remove the section matching 'targetTitle' under a specific PDF.

2.1.5 - updateUserNotes(self, username, notedata)

Extracts the 'pdfID' from 'notedata' which contains information about the notes being handled. It then accesses the 'users' collection and searches for a given username. If the user exists, the 'user' variable will contain all their data, otherwise it is empty. The method then iterates over the user's notes, and checking if there are any notes for a specific 'pdfID.' Next the chapter title is updated, existing sections are cleared to ensure outdated information is removed before new data is added. Then, the method iterated over 'notedata' and adds them to the database using the '$push' operato. If there are no notes for a user, it just adds them without deleting any old notes. If the user does not exist, a new one is created.

2.1.6 - __init__(self, username, password) - User class

Adds the username and password to make it accessible in any instance of the User class.

2.17 - getNotes(self, username, pdfID)

Retrieves notes for a given user that are associated with a specific 'pdfID' from the MongoDB database.

2.1.8 - __init__(self, pdfID, chapterTitle) - notesDS class

Assigns the pdfID, chapterTitle, to any notesDS class instance, and creates a 'self.sections' list to contain the sections created by the user

2.1.9 - addSection(self, sectionTitle, sectionNotes)

Adds a section with a corresponding title to the sections list.

2.2. TKinter - main.py

Imports:

- Tkinter
- fitz
- tempfile
- os

- sys

This file handles the creation of the Interactive Learning Module (ILB) for the ARA application using TKinter and the tkPDFViewer library. It creates what the users will be interacting with while using our software

2.2.1 - __init__(self, master, username, pdfid, db, notes_app) - NotesSection Class

Sets up the main-frame, user related details, and the database connection. It initializes a text entry widget for the section title, adds a text area for writing notes, a toggle button to collapse or expand the text area, and a delete button for removing the note section.

2.2.2 - on_text_entry(self, event)

When the user starts taking notes, the display prompts for the ‘recite’ SQ3R step.

2.2.3 - toggle(self)

Collapses the text area if it’s currently visible, and changes the toggle button text to ‘+’, or expands it if hidden, changing the text back to ‘-’.

2.2.4 - on_title_entry_click(self, event)

Clears the default title text from the entry widget when it’s clicked. If the title text is present, it sets it to black.

2.2.5 - on_title_focus_out(self, event)

Resets the title entry field to the default text and color.

2.2.6 - @property title(self)

Returns the currently entered title from the entry text box, or “Untitled section” if left empty.

2.2.7 - delete_section(self, root)

Checks for a database connection. If so, triggers deletion of the section from the database based off of a username, pdfID, and section title. It removes the section’s frame from the UI and closes the popup.

2.2.8 - on_delete_click(self)

Opens a popup confirming that a user wants to delete the section they specified. Provides ‘yes’ and ‘no’ buttons with according functionality.

2.2.9 - __init__(self, master, pdf_path, username, pdf_id, root, db)

Sets up the essential attributes like the main application window, the root window, path to the PDF document, and initializes the PDF document using the ‘fits.open’ method from the PyMuPDF library. It prepares the section list to store note sections that are created during the application’s use, and ensures that there is a database connection.

2.2.10 - load_notes(self)

Creates a ‘User’ object with the current username.It retrieves notes from the database for a given user and PDF, then populates the UI.

2.2.11 - create_notes_frame(self)

Creates a section for note taking on the right side of the main ARA window. It sets up a section for entering a chapter title with placeholder text.

2.2.12 - on_entry_click(self, event)

Event handler for the chapter title entry.

2.2.13 - on_focusout(self, event)

Event handler for clicking out of the chapter title entry widget.

2.2.14 - create_pdf_viewer(self)

Sets up a canvas for displaying the pages of the PDF, initializing it to the first page being shown.

2.2.15 - display_page(self, page_number)

Loads a specific page of the PDF, converting it into a pixmap, temporarily saves it to a file, loads the image to the canvas, then deletes the temporary file.

2.2.16 - get_current_pdf_id(self)

Returns the PDF ID currently being viewed.

2.2.17 - switch_pdf(self)

Confirms with the user that they want to switch PDFs. If so, it triggers actions that put the PDF into the current session.

2.2.18 - create_buttons(self)

Initializes and places several functional buttons throughout the ILM.

2.2.19 - toggle_prompts(self)

Toggles the visibility in the ILM of the SQ3R prompts

2.2.20 - update_prompt(self, action)

Updates the prompt based on the associated action. Switches the prompts and displays the appropriate one

2.2.21 - create_prompts(self)

This holds the text for the prompts themselves. Starts out displaying the ‘survey’ prompt first.

2.2.22 - hide_show_pdf(self)

Toggles the visibility of the PDF viewer on the canvas.

2.2.23 - next_page(self) and prev_page(self)

Moves to the next or previous page in the displayed PDF. After looking through 2 pages, the question prompt is displayed. After the user starts reading, it displays the reading prompt. If the PDF is on the first page, the ‘survey’ prompt is displayed.

2.2.24 - add_new_section(self)

Initializes a new ‘NoteSection’ object, which represents a section of notes, then adds it to the database. When the user starts to take notes, it displays the ‘recite’ prompt.

2.2.25 - add_example_notes(self)

Loops through a dictionary of sections and notes associated with chapter 2. This function is only called if the user is accessing the example_pdf . While loop through the dictionary, it adds the sections and notes for the example PDF.

2.2.26 - add_section(self, title, content)

Sets the title and content of the new section based on provided parameters. Populates the UI with the data fetched from MongoDB.

2.2.27 - save_notes(self)

Checks that the database is connected, and if so, compiles all the notes sections into a ‘notesDS’ object and updates them into the database.

2.2.28 - on_exit_buitton_click(self)

Prompts the user to either logout or quit the application with a popup window, after they click the exit button. Before quitting, the user is prompted to ‘review’ their work, the final step in SQ3R.

2.2.29 - logout(self, popup)

Saves any current notes, hides the main window, destroys the logout popup, and opens a global login screen.

2.2.30 - quit_program(self, popup)

Once the user confirms they want to leave the program, the notes are autosaved and the program is suspended.

2.2.31 - open_login_screen(root)

Manages the login through a GUI, allows selection of predefined users, including an admin user

2.2.32 - user_pdf_selection(username, root)

Provides a selection interface so the user can select one of our provided PDFs

2.2.33 - main_window(username, pdf_id, root)

Initializes the main application window to the correct size. A database object is created and the paths to the PDFs are specified

2.2.34 - main()

Serves as the entry point for the ARA. Creates the root window and hides it once the login screen is displayed