1.INTRODUCTION

A chatbot is a computer program designed to simulate conversation with human users, especially over the Internet. It is a virtual robot assistant that help human to get answers for their queries. Chatbots have become increasingly popular in recent years, particularly in the fields of education use, business and customer service. For schools and colleges, chatbots can provide students with quick and convenient access to information about their programs, courses, exam schedules and other resources. By answering common questions about course offerings, admission requirements, and other topics, chatbots can help students make informed decisions about their education. In businesses, chatbots are often used as a customer service tool, allowing customers to get answers quickly and easily to their questions. By automating routine tasks, chatbots can free up customer service representatives to focus on more complex tasks, improving the overall customer experience. This study explores the problem of exam schedules at the university by designing a computer-based application to generate an automated exam schedule that is implemented on data.

They are usually capable of producing a good enough solution for practical purposes. Solving real-world university examination timetabling has been described as a challenging research area because of various constraints that regulate the timetabling task These constraints can be classified into hard and soft constraints. Hard constraints are satisfied under any situation, while soft constraints are not essential and could be implemented as far as possible. Creating a satisfactory exam schedule manually is very difficult. While simple algorithms performing an exhaustive search are not an option in real-world cases owing to the complexity of the problem, Evolutionary algorithms (EA) based on meta-heuristics are especially well suited for solving these types of problems. EA is a class of heuristics search techniques that work iteratively on the population of candidate solutions to a given problem. Overall, chatbots are an innovative solution for organizations looking to streamline communication and improve efficiency.

2.EXISTING SYSTEM

In the existing system, students and parents usually have to wait in long lines or make multiple calls to get the exam schedule that students need. This process can result in frustration to find student's exam schedule. The manual process also means that there is no way to track or analyze the data on student's schedule inquiries, making it difficult for colleges to understand their audience and improve their services. the problem definition of the examination timetabling could be more complex in most universities or large academic institutions because of the wide variety of scheduling constraints and the vast amount of variables that need to be analyzed and considered. Despite the complexity of the examination timetabling problems and processes, many academic institutions still use manual systems (paper-based or software that does not aid the human scheduler) as their scheduling method. It is typically a manual process, where students or parents have to visit the college campus or call the admission office for information about courses, exam schedule, fees, admission criteria, etc. This system is time-consuming, prone to errors and delays, and can be overwhelming for both students and college staff.

DISADVANTAGES

- A chatbot is not a human agent.
- Drawbacks of the chatbot are the number of stored queries, it can resolve. At a certain point in time, it will have to connect to an actual human to resolve the issues.
- When a user asks a question that is not in the chatbot's database. The chatbot is unable to respond.
- The administrator should update the schedule every examination period.
- Changes on the system such as adding, editing or deleting of data can be monitored by the system.

3.PROPOSED SYSTEM

In proposed system, an exam schedule chatbot that uses natural language processing and machine learning algorithms to provide instant answers to student and parent inquiries. This system application examination scheduling system as an alternative to the college department's existing examination scheduling method and technology. It will incorporate both the concept of interactive and automated computer-based scheduling methods The chatbot can handle multiple inquiries at once, reducing the workload on college professors. They are providing students and parents with the information they need quickly and efficiently. They has the advantage of collecting data on student questions and teacher question. The chatbot is user-friendly and accessible, making it easier for students and parents to get the information they need, even if they are not familiar with the college website. The chatbot is a powerful tool that can help colleges improve their services and streamline the information-gathering process for students, teachers, and parents.

ADVANTAGES

- Chatbots are available 24 hours daily to help the students and teachers.
- They answer all the users' queries.
- They are virtual assistants for students and teachers.
- The administrator should update the schedule every examination period.
- Changes on the system such as adding, editing or deleting of data can be monitored by the system.

4.SYSTEM SPECIFICATIONS

4.1.HARDWARE REQUIREMENTS

• Processor	:	11th Gen Intel(R) Core(TM) i5-1135G
• Hard disk	:	520GB
• RAM	:	8 GB

4.2.SOFTWARE REQUIREMENTS

• Operating System	:	Windows 10/11
• Coding Language	:	Python, Tkinter, PIP packages
• Libraries	:	NLTK
• IDE	:	Anaconda and Visual studio code
• Database	:	SQLite3

5.SOFTWARE ENVIRONMENT

A software is a set of instructions and data that tell a computer what to do and how to do it. In other words, software is the intangible part of a computer that can be manipulated and interacted with by users.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was created in the late 1980s, and first released in 1991, by Guido van Rossum as a successor to the ABC programming language. Python 2.0, released in 2000, introduced new features, such as list comprehensions, and a garbage collection system with reference counting, and was discontinued with version 2.7 in 2020. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible and much Python 2 code does not run unmodified on Python 3.

PYTHON OVERVIEW

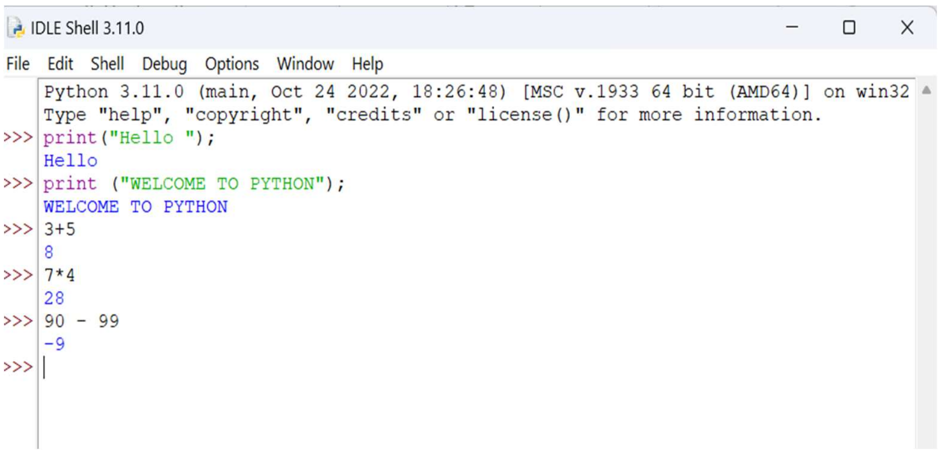
Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python’s elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python web site, https://www.python.org/, and may be freely distributed.



```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello ");
Hello
>>> print ("WELCOME TO PYTHON");
WELCOME TO PYTHON
>>> 3+5
8
>>> 7*4
28
>>> 90 - 99
-9
>>> 
```

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

HISTORY:

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL,capable of exception handling and interfacing with the Amoeba operating system.[9] Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator For Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. He now shares his leadership as a member of a five-person steering council. In January 2019, active Python core developers elected Brett Cannon, Nick Coghlan, Barry Warsaw, Carol Willing and Van Rossum to a five-member "Steering Council" to lead the project.

Guido van Rossum has since then withdrawn his nomination for the 2020 Steering council. Python 2.0 was released on 16 October 2000 with many major new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible.Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2to3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.

DESIGN PHILOSOPHY AND FEATURES

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)).Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets, and generator expressions.

The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.

- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

STATEMENTS AND CONTROL FLOW

Python's statements include (among others):

- The assignment statement (token '=', the equals sign).
- The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
- The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
- The while statement, which executes a block of code as long as its condition is true.
- The try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.

METHODS

Methods on objects are functions attached to the object's class; the syntax instance. method(argument) is, for normal methods and functions, syntactic sugar for Class. method(instance, argument). Python methods have an explicit self parameter to access instance data, in contrast to the implicit self (or this) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby).

TYPING

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically-typed, Python is strongly-typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows programmers to define their own types using classes, which are most often used for object-oriented programming. New instances of classes are constructed by calling the class (for example, SpamClass() or EggsClass()), and the classes are instances of the metaclass type (itself an instance of itself), allowing metaprogramming and reflection.Before version 3.0, Python had two kinds of classes: old-style and new-style. The syntax of both styles is the same, the difference being whether the class object is inherited from, directly or indirectly (all new-style classes inherit from object and are instances of type).

In versions of Python 2 from Python 2.2 onwards, both kinds of classes can be used. Old-style classes were eliminated in Python 3.0. The long-term plan is to support gradual typing and from Python 3.5, the syntax of the language allows specifying static types but they are not checked in the default implementation, CPython. An experimental optional static type checker named mypy supports compile-time type checking.

PYTHON INSTALL

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

To check if you have python installed on a Linux or Mac, then on Linux open the command line or on Mac open the Terminal and type:

```
python --version
```

If you find that you do not have Python installed on your computer, then you can download it for free from the following website: https://www.python.org/

PYTHON QUICKSTART

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

```
C:\Users\Your Name>python helloworld.py
```

Where "helloworld.py" is the name of your python file.
Let's write our first Python file, called helloworld.py, which can be done in any text editor.

THE PYTHON COMMAND LINE

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.
Type the following on the Windows, Mac or Linux command line:

```
C:\Users\Your Name>python
```

Or, if the "python" command did not work, you can try "py":

```
C:\Users\Your Name>py
```

From there you can write any python, including our hello world example from earlier in the tutorial:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)]
on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
```

Which will write "Hello, World!" in the command line:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)]
on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>print("Hello, World!")
Hello, World!
```

LIBRARIES

1. **Tkinter :** The Tkinter library is a built-in Python library used to create graphical user interfaces (GUIs). It provides a wide range of widgets such as buttons, labels, text boxes, and frames that can be used to design interactive applications. Tkinter is a cross-platform library that works on various operating systems such as Windows, macOS, and Linux. It is based on the Tk GUI toolkit and provides a Python interface for it. Tkinter provides a simple and easy-to-use interface for designing GUIs, making it a popular choice for beginners and experienced developers. With its user-friendly interface and powerful features, Tkinter has become a popular choice for developing GUI applications in Python.

   ```
   import tkinter as tk
   ```

2. **NLTK:** The Natural Language Toolkit (nltk) is a popular Python library for working with human language data. It provides a suite of tools and resources for tasks such as tokenization, stemming, tagging, parsing, and classification of natural language text. The nltk library contains many useful features and datasets, including corpora of text in various languages, lexical resources such as WordNet, and algorithms for natural language processing tasks such as part-of-speech tagging and named entity recognition.

   ```
   from nltk.chat.util import Chat, reflections
   ```

3. **Prettytable :** PrettyTable is a Python library that allows creating and displaying tables in a visually appealing manner. It is built on top of the standard Python library, making it easy to use and install. The library allows generating tables from data in various formats, such as CSV, JSON, or Python objects. The tables generated by the library can be customized with various formatting options, such as setting column widths, changing the alignment of text, or setting the font size.

```
from prettytable import PrettyTable, prettytable.
```

USES

Since 2003, Python has consistently ranked in the top ten most popular programming languages in the TIOBE Programming Community Index where, as of February 2020, it is the third most popular language (behind Java, and C). It was selected Programming Language of the Year in 2007, 2010, and 2018. An empirical study found that scripting languages, such as Python, are more productive than conventional languages, such as C and Java, for programming problems involving string manipulation and search in a dictionary, and determined that memory consumption was often "better than Java and not much worse than C or C++".Large organizations that use Python include Wikipedia, Google, Yahoo!, CERN, NASA,Facebook, Amazon, Instagram, Spotify and some smaller entities like ILM and ITA.

The social news networking site Reddit is written entirely in Python.Python can serve as a scripting language for web applications, e.g., via mod_wsgi for the Apache web server. With Web Server Gateway Interface, a standard API has evolved to facilitate these applications. Web frameworks like Django, Pylons, Pyramid, TurboGears, web2py, Tornado, Flask, Bottle and Zope support developers in the design and maintenance of complex applications. Pyjs and IronPython can be used to develop the client-side of Ajax-based applications. SQLAlchemy can be used as a data mapper to a relational database. Twisted is a framework to program communications between computers, and is used (for example) by Dropbox.

Libraries such as NumPy, SciPy and Matplotlib allow the effective use of Python in scientific computing, with specialized libraries such as Biopython and Astropy providing domain-specific functionality. SageMath is a mathematical software with a notebook interface programmable in Python: its library covers many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus.OpenCV has python bindings with a rich set of features for computer vision and image processing.

TKINTER

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python.

WINDOW

This term has different meanings in different contexts, but in general it refers to a rectangular area somewhere on the user's display screen.

TOP-LEVEL WINDOW

A window which acts as a child of the primary window. It will be decorated with the standard frame and controls for the desktop manager. It can be moved around the desktop and can usually be resized.

WIDGET

The generic term for any of the building blocks that make up an application in a graphical user interface.

- Core widgets: The containers: frame, labelframe, toplevel, paned window. The buttons: button, radiobutton, checkbutton (checkbox), and menubutton. The text widgets: label, message, text. The entry widgets: scale, scrollbar, listbox, slider, spinbox, entry (singleline), optionmenu, text (multiline), and canvas (vector and pixel graphics).
- Tkinter provides three modules that allow pop-up dialogs to be displayed: tk.messagebox (confirmation, information, warning and error dialogs), tk.filedialog (single file, multiple file and directory selection dialogs) and tk.colorchooser (colour picker).
- Python 2.7 and Python 3.1 incorporate the "themed Tk" ("ttk") functionality of Tk 8.5. This allows Tk widgets to be easily themed to look like the native desktop environment in which the application is running, thereby addressing a long-standing criticism of Tk (and hence of Tkinter). Some widgets are exclusive to ttk, such as the combobox, progressbar and treeview widgets.

FRAME

In Tkinter, the Frame widget is the basic unit of organization for complex layouts. A frame is a rectangular area that can contain other widgets.

CHILD AND PARENT

When any widget is created, a parent–child relationship is created. For example, if you place a text label inside a frame, the frame is the parent of the label.

ANACONDA

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions in Anaconda are managed by the package management system conda.

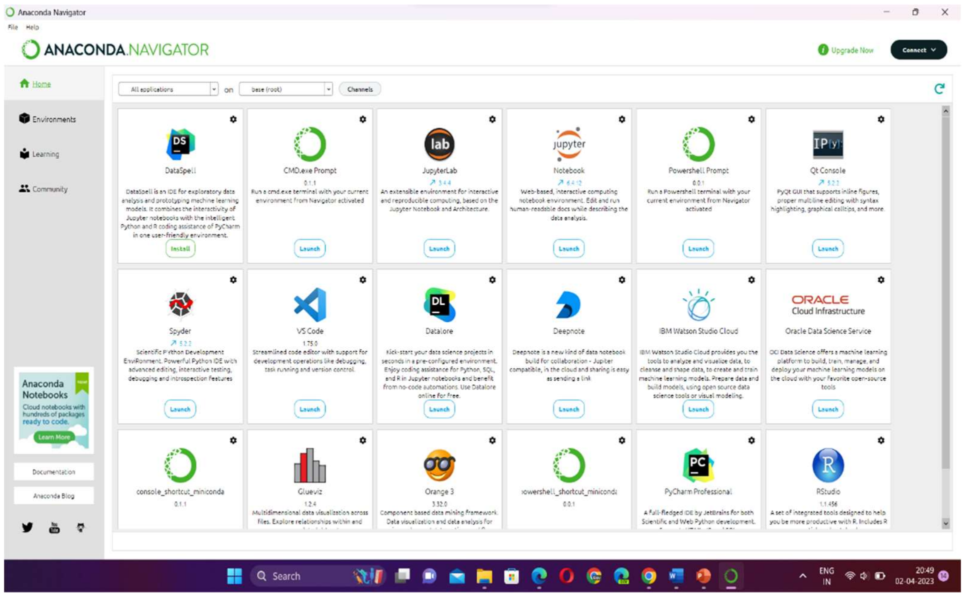
Anaconda Distribution is a Python/R data science distribution and a collection of over 7,500+ open-source packages, which includes a package and environment manager. Anaconda Distribution is platform-agnostic, so you can use it whether you are on Windows, macOS, or Linux.

Anaconda Navigator is an application provided with Anaconda Distribution that enables users to work with conda packages and environments using a graphical user interface (GUI) instead of a command line interface (CLI), like Anaconda Prompt on Windows or a terminal.

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The CLI program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is a graphical interface that enables you work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.



DataSpell
DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.

CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

JupyterLab
3.4.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.4.12
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated

Qt Console
5.2.2
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
5.2.2
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

VS Code
1.75.0
Streamlined code editor with support for development operations like debugging, task running and version control.

Datalore
Kick-start your data science projects in seconds in a pre-configured environment. Enjoy coding assistance for Python, SQL, and R in Jupyter notebooks and benefit from no-code automations. Use Datalore online for free.

Deepnote
Deepnote is a new kind of data notebook build for collaboration - Jupiter compatible, in the cloud and sharing is easy as sending a link

IBM Watson Studio Cloud
IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

ORACLE Cloud Infrastructure
Oracle Data Science Service
OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source tools

console_shortcut_miniconda
0.1.1

Glueviz
1.2.4
Multidimensional data visualization across files. Explore relationships within and

Orange 3
3.32.0
Component based data mining framework. Data visualization and data analysis for

owershell_shortcut_minicond
0.0.1

PyCharm Professional
A full-fledged IDE by JetBrains for both Scientific and Web Python development.

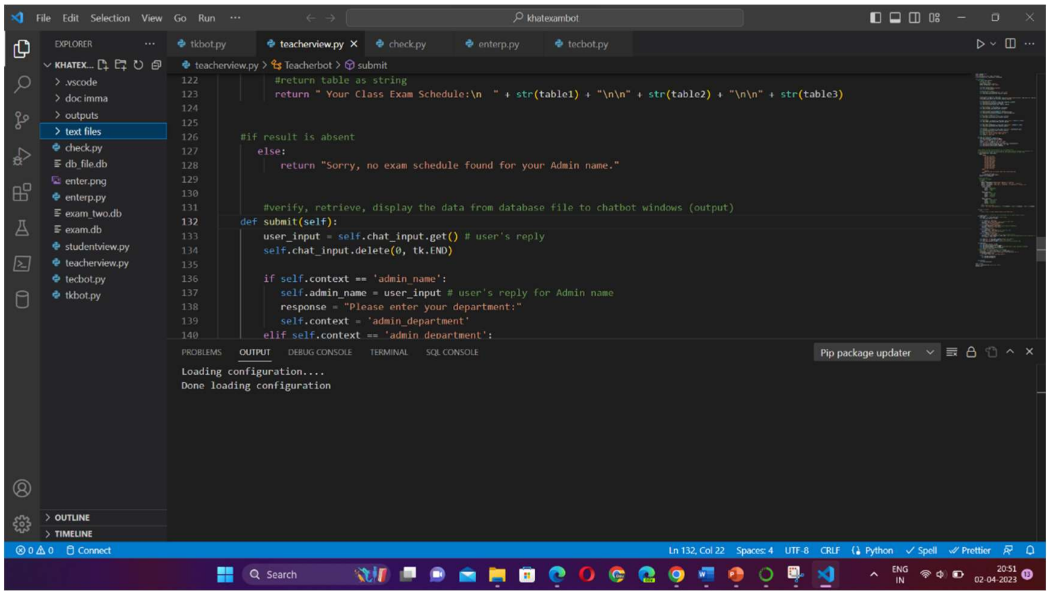
RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R

VISUAL STUDIO CODE OVERVIEW

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

Visual Studio Code is a free, open-source code editor developed by Microsoft. It is a lightweight but powerful editor that provides features like debugging, IntelliSense, Git integration, and extensions. The software runs on Windows, macOS, and Linux. Programming languages supported by VS Code are C++ - C# - CSS - Dart - Docker file - F# - Go - HTML - Java - JavaScript - JSON - Julia - Less - Markdown - PHP - PowerShell - Python - R - Ruby - Rust - SCSS - T-SQL - TypeScript.

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.



```python
            #return table as string
            return " Your Class Exam Schedule:\n  " + str(table1) + "\n\n" + str(table2) + "\n\n" + str(table3)


    #if result is absent
      else:
          return "Sorry, no exam schedule found for your Admin name."


        #verify, retrieve, display the data from database file to chatbot windows (output)
    def submit(self):
        user_input = self.chat_input.get() # user's reply
        self.chat_input.delete(0, tk.END)

        if self.context == 'admin_name':
            self.admin_name = user_input # user's reply for Admin name
            response = "Please enter your department:"
            self.context = 'admin_department'
        elif self.context == 'admin_department':
```

```
Loading configuration....
Done loading configuration
```

SQLITE3

A C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world. SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day.

The SQLite file format is stable, cross-platform, and backwards compatible and the developers pledge to keep it that way through the year 2050. SQLite database files are commonly used as containers to transfer rich content between systems. The database (.db) file is file format of SQLite3 that used in python program.

SQLite is a "light" version that works over syntax very much similar to SQL. SQLite is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine. It is the most used database engine on the world wide web. Python has a library to access SQLite databases, called sqlite3, intended for working with this database which has been included with Python package since version 2.5. SQLite has the following features.

1. Serverless
2. Self-Contained
3. Zero-Configuration

4. Transactional
5. Single-Database

SQLITE QUICKSTART

Start the **sqlite3** program by typing "sqlite3" at the command prompt, optionally followed by the name of the file that holds the SQLite database. If the named file does not exist, a new database file with the given name will be created automatically. If no database file is specified on the command-line, a temporary database is created and automatically deleted when the "sqlite3" program exits.

On startup, the **sqlite3** program will show a brief banner message then prompt you to enter SQL. Type in SQL statements (terminated by a semicolon), press "Enter" and the SQL will be executed.

This is the example to create a new SQLite database named "ex1" with a single table named "tbl1", you might do this:

```
$ sqlite3 ex1
SQLite version 3.36.0 2021-08-03 18:36:39
Enter ".help" for usage hints.
sqlite> create table stl1(one text, two int);
sqlite> insert into stl1 values('hello!',10);
sqlite> insert into stl1 values('goodbye', 20);
sqlite> select * from stl1;
hello!|10
goodbye|20
sqlite>
```

PYTHON SQLITE3 MODULE APIs

Following are important sqlite3 module routines, which can suffice your requirement to work with SQLite database from your Python program. If you are looking for a more sophisticated application, then you can look into Python sqlite3 module's official documentation.

Sr.No.	API & Description
1	**sqlite3.connect(database [,timeout ,other optional arguments])** This API opens a connection to the SQLite database file. You can use ":memory:" to open a database connection to a database that resides in RAM instead of on disk. If database is opened successfully, it returns a connection object. When a database is accessed by multiple connections, and one of the processes modifies the database, the SQLite database is locked until that transaction is committed. The timeout parameter specifies how long the connection should wait for the lock to go away until raising an exception. The default for the timeout parameter is 5.0 (five seconds). If the given database name does not exist then this call will create the database. You can specify filename with the required path as well if you want to create a database anywhere else except in the current directory.
2	**connection.cursor([cursorClass])** This routine creates a **cursor** which will be used throughout of your database programming with Python. This method accepts a single optional parameter cursorClass. If supplied, this must be a custom cursor class that extends sqlite3.Cursor.
3	**cursor.execute(sql [, optional parameters])** This routine executes an SQL statement. The SQL statement may be parameterized (i. e. placeholders instead of SQL literals). The sqlite3 module supports two kinds of placeholders: question

	marks and named placeholders (named style). **For example** − cursor.execute("insert into people values (?, ?)", (who, age))
4	**connection.execute(sql [, optional parameters])** This routine is a shortcut of the above execute method provided by the cursor object and it creates an intermediate cursor object by calling the cursor method, then calls the cursor's execute method with the parameters given.
5	**cursor.executemany(sql, seq_of_parameters)** This routine executes an SQL command against all parameter sequences or mappings found in the sequence sql.
6	**connection.executemany(sql[, parameters])** This routine is a shortcut that creates an intermediate cursor object by calling the cursor method, then calls the cursor.s executemany method with the parameters given.
7	**cursor.executescript(sql_script)** This routine executes multiple SQL statements at once provided in the form of script. It issues a COMMIT statement first, then executes the SQL script it gets as a parameter. All the SQL statements should be separated by a semi colon (;).
8	**connection.executescript(sql_script)** This routine is a shortcut that creates an intermediate cursor object by calling the cursor method, then calls the cursor's executescript method with the parameters given.

9	**connection.total_changes()** This routine returns the total number of database rows that have been modified, inserted, or deleted since the database connection was opened.
10	**connection.commit()** This method commits the current transaction. If you don't call this method, anything you did since the last call to commit() is not visible from other database connections.
11	**connection.rollback()** This method rolls back any changes to the database since the last call to commit().
12	**connection.close()** This method closes the database connection. Note that this does not automatically call commit(). If you just close your database connection without calling commit() first, your changes will be lost!
13	**cursor.fetchone()** This method fetches the next row of a query result set, returning a single sequence, or None when no more data is available.
14	**cursor.fetchmany([size = cursor.arraysize])** This routine fetches the next set of rows of a query result, returning a list. An empty list is returned when no more rows are available. The method tries to fetch as many rows as indicated by

	the size parameter.
15	**cursor.fetchall()** This routine fetches all (remaining) rows of a query result, returning a list. An empty list is returned when no rows are available.

SQLITESTUDIO

The SQLiteStudio tool is a free GUI tool for managing SQLite databases. It is free, portable, intuitive, and cross-platform. SQLite tool also provides some of the most important features to work with SQLite databases such as importing, exporting data in various formats including CSV, XML, and JSON.

You can download the SQLiteStudio installer or its portable version by visiting the download page. Then, you can extract (or install) the download file to a folder e.g., C:\sqlite\gui\ and launch it. The following picture illustrates how to launch the SQLiteStudio:

Install SQLite GUI tool

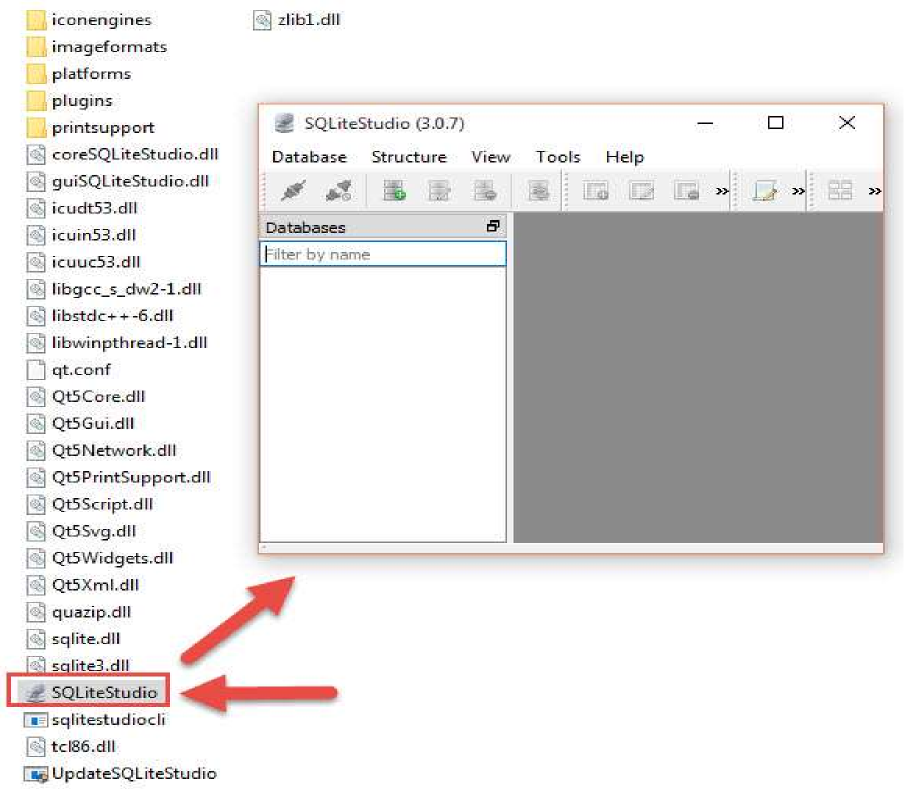
The sqlite3 shell is excellent. However, sometimes, you may want to work with the SQLite databases using an intuitive GUI tool. There are many GUI tools for managing SQLite databases available ranging from freeware to commercial licenses.

SQLiteStudio:

The SQLiteStudio tool is a free GUI tool for managing SQLite databases. It is free, portable, intuitive, and cross-platform. SQLite tool also provides some of the most important features to work with SQLite databases such as importing, exporting data in various formats including CSV, XML, and JSON.

You can download the SQLiteStudio installer or its portable version by visiting the download page. Then, you can extract (or install) the download file to a folder

e.g., C:\sqlite\gui\ and launch it. The following picture illustrates how to launch the SQLiteStudio:



Other SQLite GUI tools

Besides the SQLite Studio, you can use the following free SQLite GUI tools:

- DBeaver is another free multi-platform database tool. It supports all popular major relational database systems MySQL, PostgreSQL, Oracle, DB2, SQL Server, Sybase.including SQLite.
- DB Browser for SQLite – is an open-source tool to manage database files compatible with SQLite.

6.SYSTEM STUDY

6.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are,

- Economical Feasibility
- Technical Feasibility
- Social Feasibility

6.2 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

6.3 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

6.4 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

7.PROJECT DESCRIPTIONS

The goal of the chatbot project used to build a conversational interface which can handle a wide range of queries. They provide accurate, relevant, and personalized responses to users. The chatbot is trained using machine learning algorithms or artificial intelligence processes that enable it to learn from previous interactions and improve its responses over time.

In the education field, chatbots can be used to answer frequently asked questions about the exam schedule, fees, administration, etc. They can help reduce the workload of college administrators and provide students with quick and easy access to information. In most companies, chatbots can be used to handle customer queries, provide product information, and even process orders. This can improve the overall customer experience and help businesses save time and resources.

7.1 OBJECTIVES

1. **Natural Language Processing (NLP)**: Chatbots are equipped with NLP algorithms, which allow them to understand and respond to human language in a conversational manner.
2. **Automated Responses**: Chatbots can provide automated responses to common questions and requests, reducing the need for human interaction.
3. **Multi-channel Integration**: Chatbots can be integrated into various communication channels, including social media platforms, messaging apps, and websites.
4. **Customization**: Chatbots can be customized to meet the specific needs of a business or organization, including the ability to add new responses and functions as needed.
5. **Scalability**: Chatbots can handle a large volume of interactions and can scale up or down to meet changing customer demands.
6. **Data Collection and Analysis**: Chatbots can collect data on customer interactions, which can be analysed to gain insights into customer behaviour and preferences. This information can be used to improve the chatbot's functionality and overall user's experience.

8.MODULES DESCRIPTION

MODULES

- **PROCESS**
- **STUDENT BOT**
- **TEACHER BOT**
- **WORK MODULES**

MODULES DESCRIPTION

PROCESS

- Chatbot has user interaction, automatic response, collects and retrieves of the data.
- Examination Scheduler is the scheduling algorithms for the manual and automatic scheduling modes.
- This component displays the different kinds of messages and output that the scheduling algorithm will produce.

STUDENT BOT

- Give the output by entering a particular student's register number, full name, class, and dob.
- They retrieve details from the database to display the exam schedule.

TEACHER BOT

- Give the output by entering a particular teacher's full name, department, class, and batch year.
- They retrieve details from the database to display the exam schedule of that particular class.

WORK MODULES

- Dialogue Management: This module manages the conversation flow between the chatbot and the user. It helps the chatbot to understand the context and respond appropriately by selecting the right information from the database.

- Intent Recognition: This module is responsible for recognizing the user's intent and mapping it to a specific action or response. The module uses machine learning algorithms to analyse the user's input and determine the most suitable action based on the context.

- Knowledge Base: This module contains all the information and data that the chatbot needs to answer the user's queries. This information can be in the form of text, images, videos, or any other type of multimedia content. The chatbot can access this information and use it to respond to the user's queries.

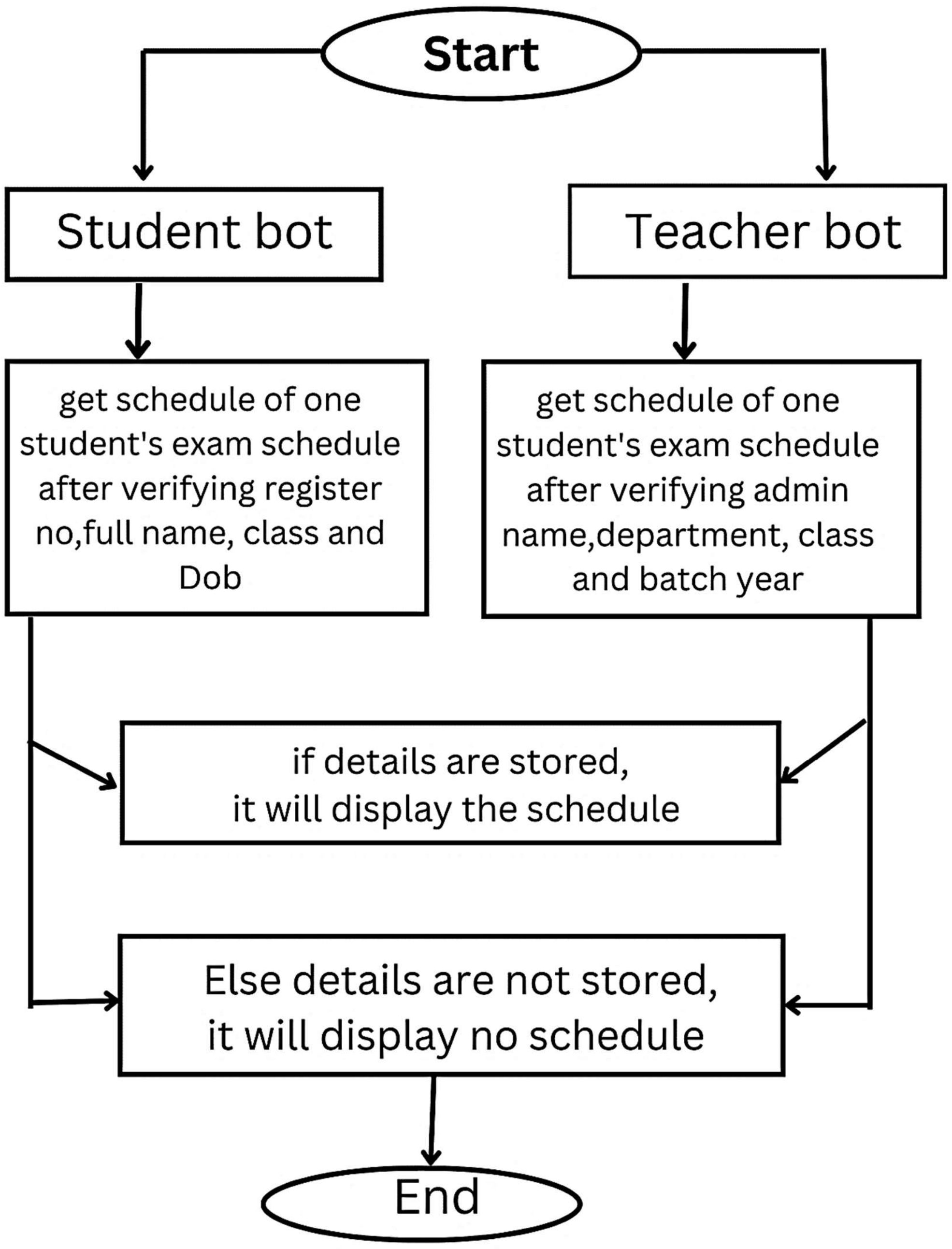
- User Interaction: This module is responsible for handling the user interface of the chatbot. It includes the design and functionality of the chatbot's interface, such as buttons, menus, and forms, and ensures that the user can interact with the chatbot in a smooth and intuitive way.

9.SYSTEM ANALYSIS AND DESIGN

9.1 DATA FLOW DIAGRAM

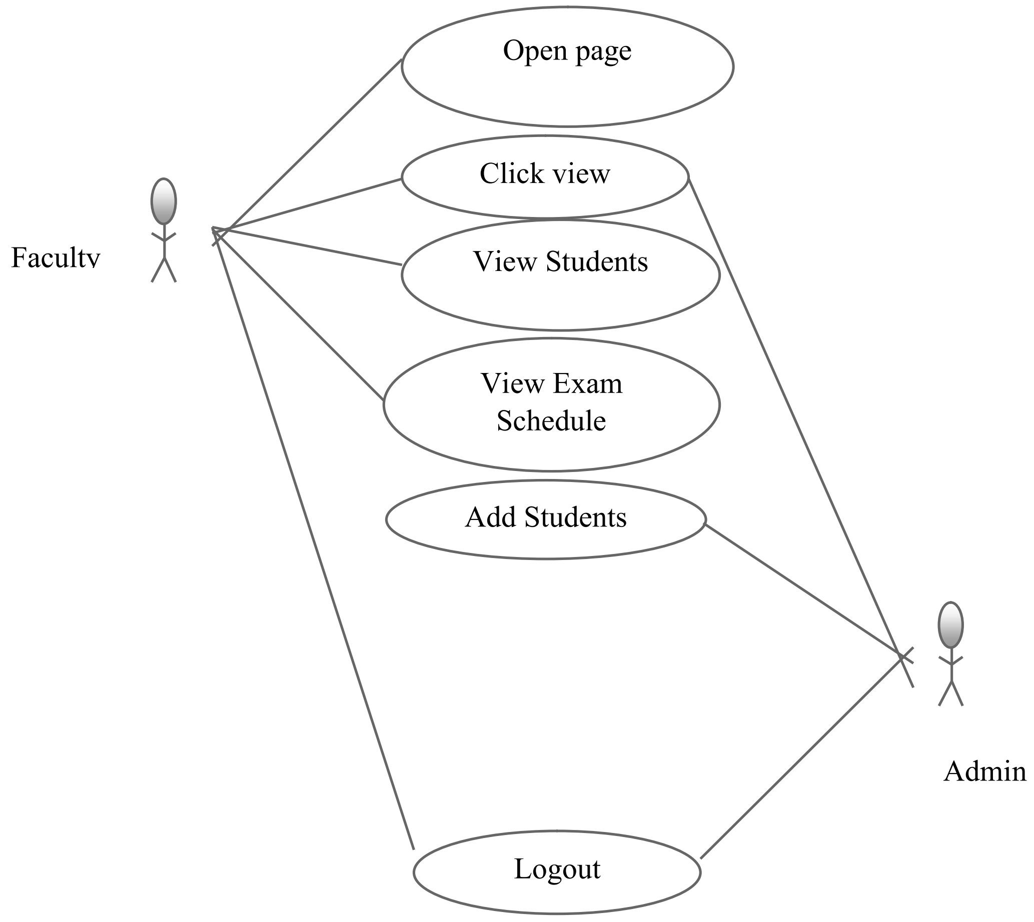
The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
The data flow diagram (DFD) is one of the most important modelling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.


Start
Student bot
Teacher bot
get schedule of one student's exam schedule after verifying register no,full name, class and Dob
get schedule of one student's exam schedule after verifying admin name,department, class and batch year
if details are stored, it will display the schedule
Else details are not stored, it will display no schedule
End

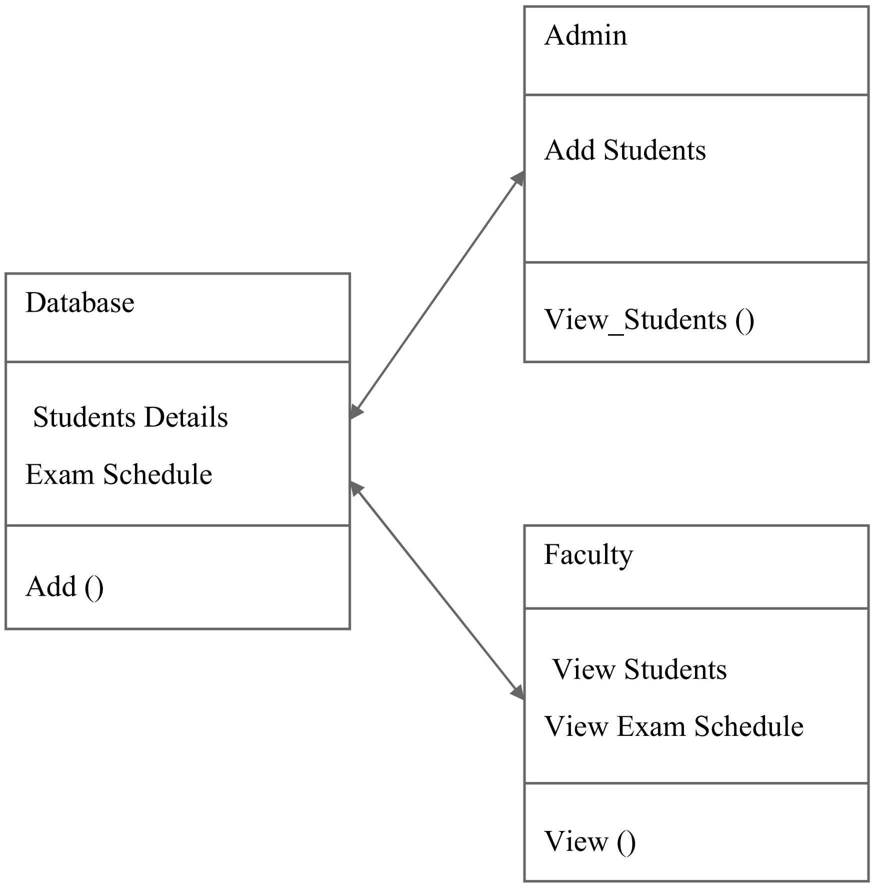
9.2 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



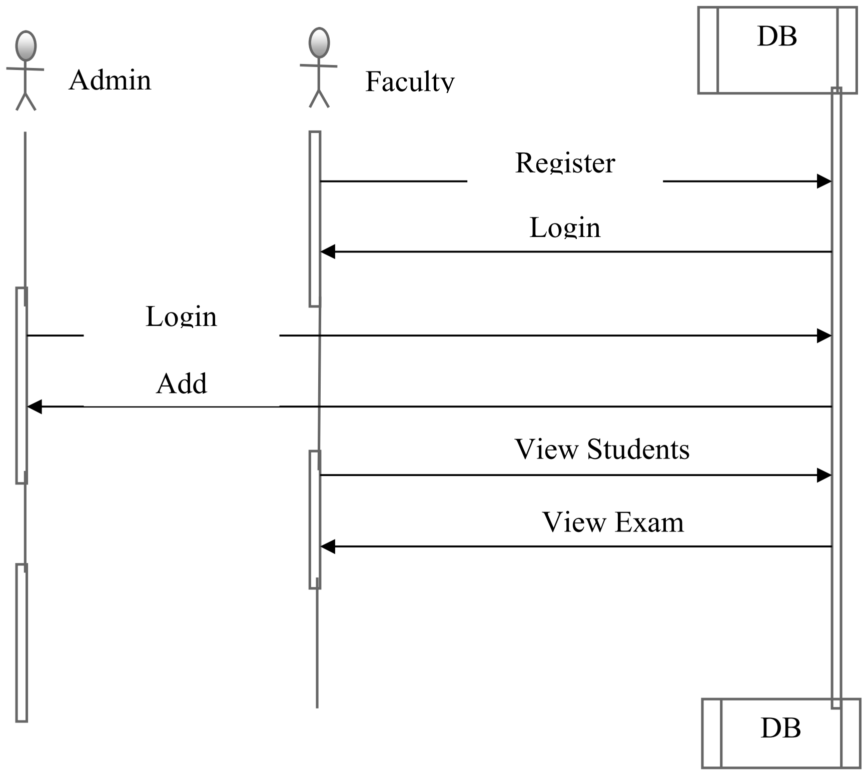
9.3 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



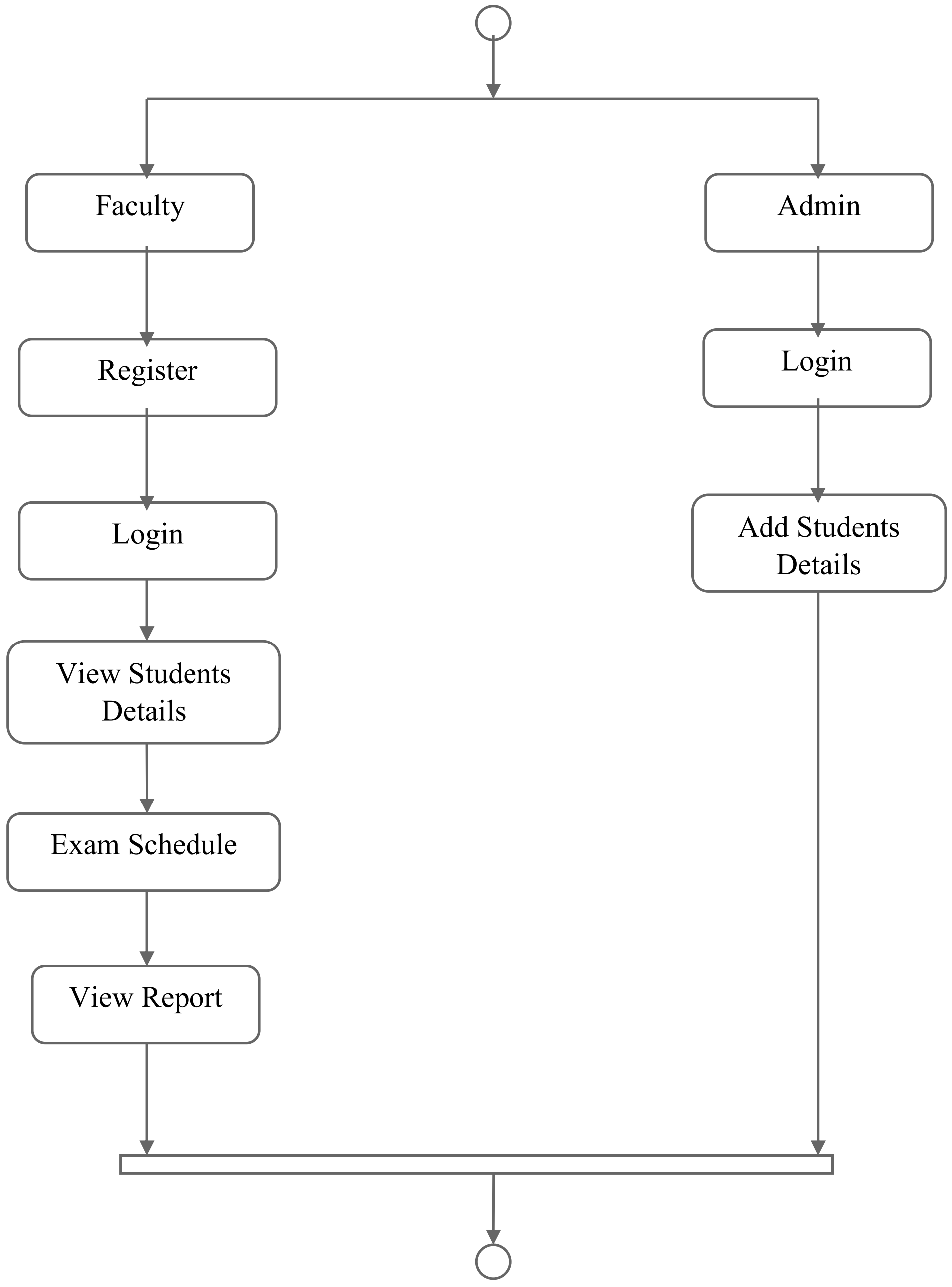
9.4 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

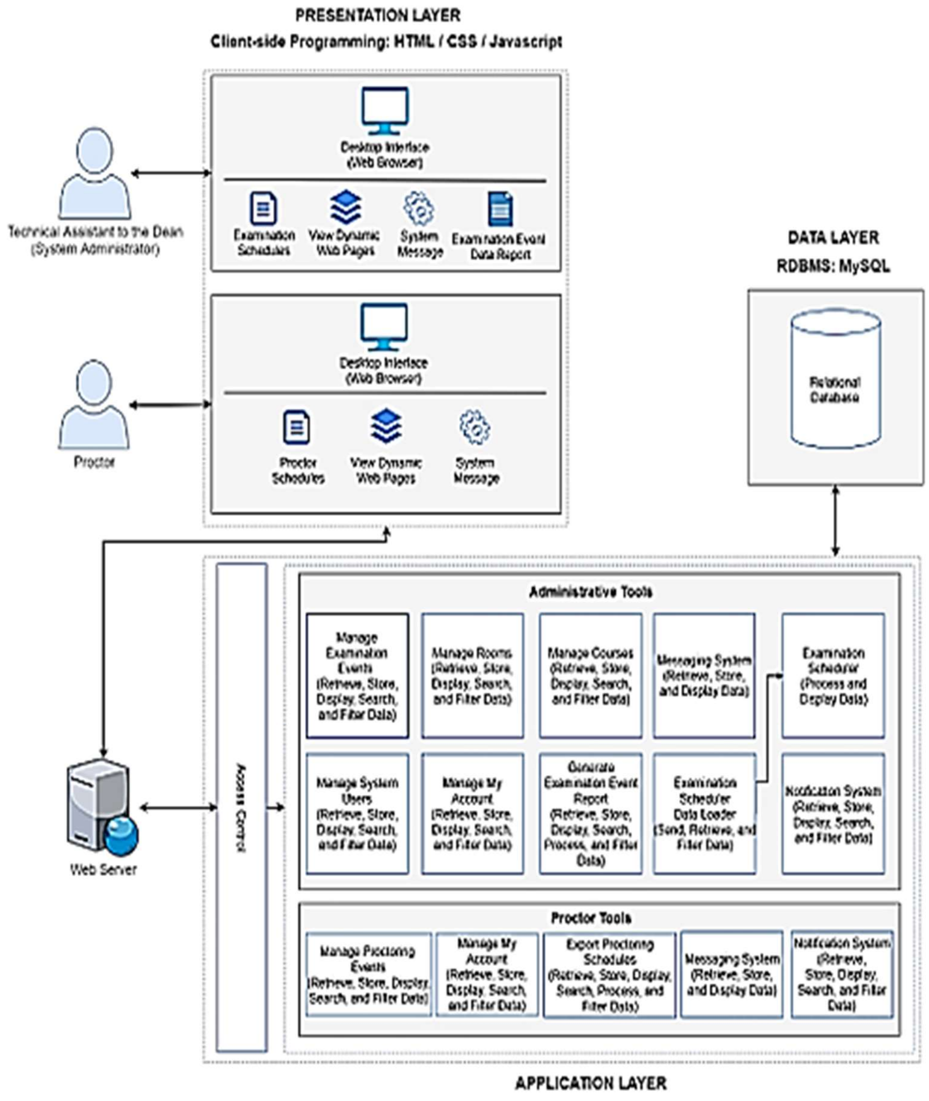


9.5 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



9.6 SYSTEM ARCHITECTURE


PRESENTATION LAYER
Client-side Programming: HTML / CSS / Javascript
Desktop Interface (Web Browser)
Examination Schedules
View Dynamic Web Pages
System Message
Examination Event Data Report
Technical Assistant to the Dean (System Administrator)
Desktop Interface (Web Browser)
Proctor Schedules
View Dynamic Web Pages
System Message
Proctor
DATA LAYER
RDBMS: MySQL
Relational Database
Web Server
Access Control
Administrative Tools
Manage Examination Events (Retrieve, Store, Display, Search, and Filter Data)
Manage Rooms (Retrieve, Store, Display, Search, and Filter Data)
Manage Courses (Retrieve, Store, Display, Search, and Filter Data)
Messaging System (Retrieve, Store, and Display Data)
Examination Scheduler (Process and Display Data)
Manage System Users (Retrieve, Store, Display, Search, and Filter Data)
Manage My Account (Retrieve, Store, Display, Search, and Filter Data)
Generate Examination Event Report (Retrieve, Store, Display, Search, Process, and Filter Data)
Examination Scheduler Data Loader (Send, Retrieve, and Filter Data)
Notification System (Retrieve, Store, Display, Search, and Filter Data)
Proctor Tools
Manage Proctoring Events (Retrieve, Store, Display, Search, and Filter Data)
Manage My Account (Retrieve, Store, Display, Search, and Filter Data)
Export Proctoring Schedules (Retrieve, Store, Display, Search, Process, and Filter Data)
Messaging System (Retrieve, Store, and Display Data)
Notification System (Retrieve, Store, Display, Search, and Filter Data)
APPLICATION LAYER

9.7 INPUT DESIGN

The input design for the exam schedule chatbot is an essential aspect of the system, providing a seamless interface between the user and the chatbot. The input design is focused on ensuring that the data entered by the user is accurate, secure, and easy to use. The input design specifications and procedures are developed to put the transaction data into a usable form for processing by the chatbot.

The input design focuses on minimizing the amount of data required, reducing errors, minimizing delays, streamlining the process, and maintaining privacy. The input design considers the following factors:

1. What information should be collected as input?
2. How should the data be arranged or coded?
3. The dialog to guide the user in providing input.
4. Methods for input validation and steps to follow if errors occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

9.8 OUTPUT DESIGN

The output design of a chatbot for inquiries is crucial in delivering relevant information to the end user. It needs to present the information in a clear and concise manner that meets the user's needs and requirements. The chatbot's output should be designed in an organized and well-thought-out manner to ensure that it is user-friendly and easy to understand.

The chatbot should have a variety of methods for presenting information, including text-based responses, visual aids, and graphical representations.

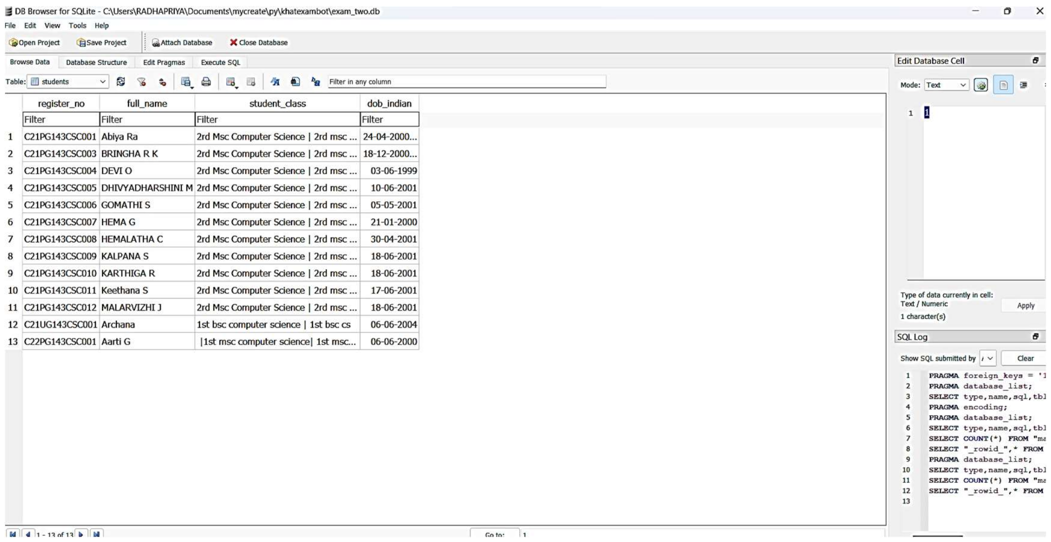
The chatbot's output should be designed to convey information about college admissions, courses offered, fees, scholarships, and other relevant topics in a straightforward and accessible way.

The chatbot's output should also be designed to signal important events, opportunities, and problems to the user. For example, if there are limited seats available for a particular course, the chatbot should be able to inform the user and guide them in making an informed decision.

The chatbot should also be designed to trigger actions and confirm actions. For example, the chatbot should be able to confirm a user's course selection or provide a link for payment of fees.

9.9 DATABASE DESIGN

STUDENT QUESTION TABLE



DB Browser for SQLite - C:\Users\RADHAPRIYA\Documents\mycreate\py\khatexambot\exam_two.db

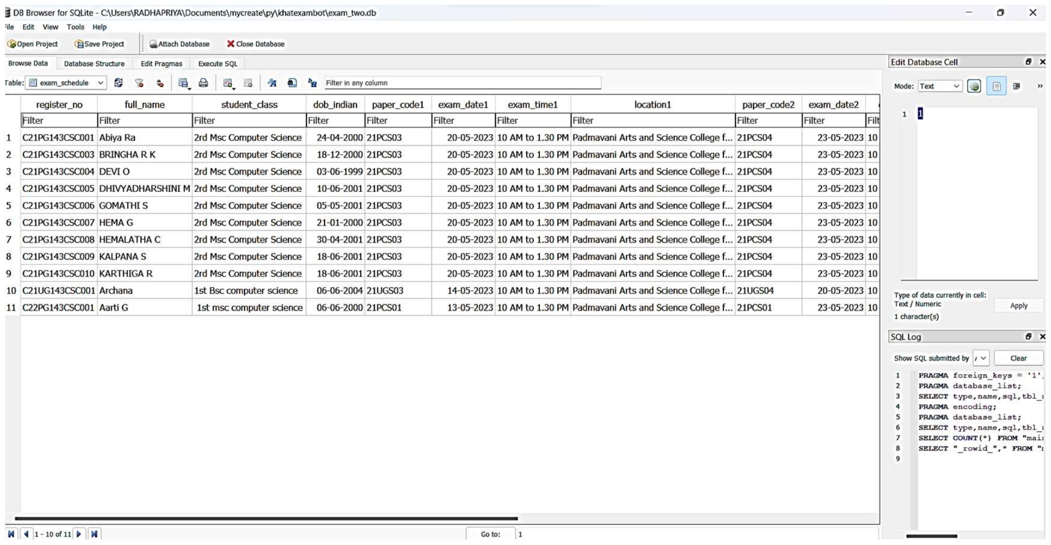
Table: students

	register_no	full_name	student_class	dob_indian
1	C21PG143CSC001	Abiya Ra	2rd Msc Computer Science | 2rd msc ...	24-04-2000...
2	C21PG143CSC003	BRINGHA R K	2rd Msc Computer Science | 2rd msc ...	18-12-2000...
3	C21PG143CSC004	DEVI O	2rd Msc Computer Science | 2rd msc ...	03-06-1999
4	C21PG143CSC005	DHIVYADHARSHINI M	2rd Msc Computer Science | 2rd msc ...	10-06-2001
5	C21PG143CSC006	GOMATHI S	2rd Msc Computer Science | 2rd msc ...	05-05-2001
6	C21PG143CSC007	HEMA G	2rd Msc Computer Science | 2rd msc ...	21-01-2000
7	C21PG143CSC008	HEMALATHA C	2rd Msc Computer Science | 2rd msc ...	30-04-2001
8	C21PG143CSC009	KALPANA S	2rd Msc Computer Science | 2rd msc ...	18-06-2001
9	C21PG143CSC010	KARTHIGA R	2rd Msc Computer Science | 2rd msc ...	18-06-2001
10	C21PG143CSC011	Keethana S	2rd Msc Computer Science | 2rd msc ...	17-06-2001
11	C21PG143CSC012	MALARVIZHI J	2rd Msc Computer Science | 2rd msc ...	18-06-2001
12	C21UG143CSC001	Archana	1st bsc computer science | 1st bsc cs	06-06-2004
13	C22PG143CSC001	Aarti G	|1st msc computer science| 1st msc...	06-06-2000

Type of data currently in cell: Text / Numeric
1 character(s)

```
PRAGMA foreign_keys = '1
PRAGMA database_list;
SELECT type,name,sql,tbl
PRAGMA encoding;
PRAGMA database_list;
SELECT type,name,sql,tbl
SELECT COUNT(*) FROM "ma
SELECT "_rowid_",* FROM
PRAGMA database_list;
SELECT type,name,sql,tbl
SELECT COUNT(*) FROM "ma
SELECT "_rowid_",* FROM
```

STUDENT ANSWER TABLE



DB Browser for SQLite - C:\Users\RADHAPRIYA\Documents\mycreate\py\khatexambot\exam_two.db

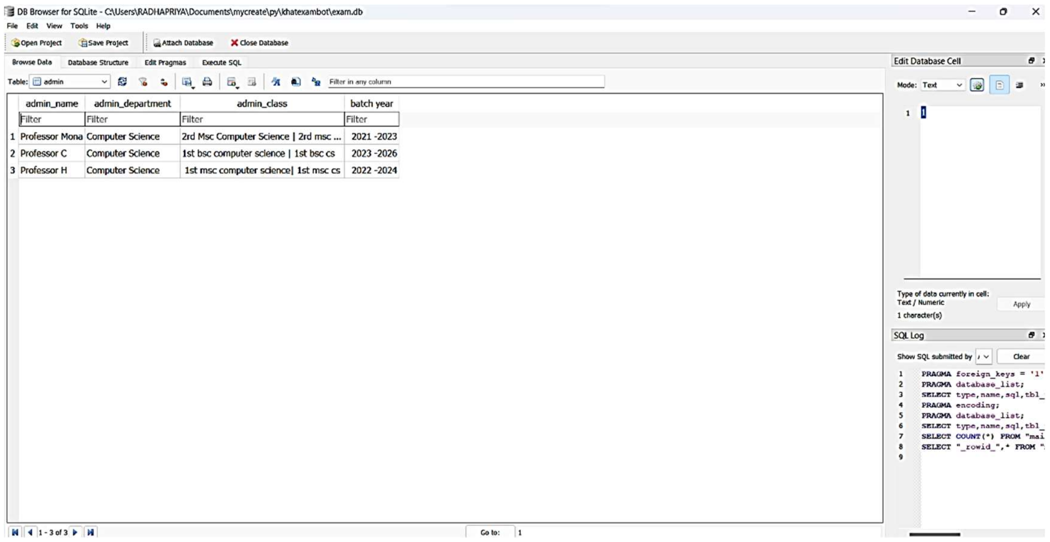
Table: exam_schedule

	register_no	full_name	student_class	dob_indian	paper_code1	exam_date1	exam_time1	location1	paper_code2	exam_date2	
1	C21PG143CSC001	Abiya Ra	2rd Msc Computer Science	24-04-2000	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College f...	21PCS04	23-05-2023	10
2	C21PG143CSC003	BRINGHA R K	2rd Msc Computer Science	18-12-2000	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College f...	21PCS04	23-05-2023	10
3	C21PG143CSC004	DEVI O	2rd Msc Computer Science	03-06-1999	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College f...	21PCS04	23-05-2023	10
4	C21PG143CSC005	DHIVYADHARSHINI M	2rd Msc Computer Science	10-06-2001	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College f...	21PCS04	23-05-2023	10
5	C21PG143CSC006	GOMATHI S	2rd Msc Computer Science	05-05-2001	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College f...	21PCS04	23-05-2023	10
6	C21PG143CSC007	HEMA G	2rd Msc Computer Science	21-01-2000	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College f...	21PCS04	23-05-2023	10
7	C21PG143CSC008	HEMALATHA C	2rd Msc Computer Science	30-04-2001	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College f...	21PCS04	23-05-2023	10
8	C21PG143CSC009	KALPANA S	2rd Msc Computer Science	18-06-2001	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College f...	21PCS04	23-05-2023	10
9	C21PG143CSC010	KARTHIGA R	2rd Msc Computer Science	18-06-2001	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College f...	21PCS04	23-05-2023	10
10	C21UG143CSC001	Archana	1st Bsc computer science	06-06-2004	21UGS03	14-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College f...	21UGS04	20-05-2023	10
11	C22PG143CSC001	Aarti G	1st msc computer science	06-06-2000	21PCS01	13-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College f...	21PCS01	23-05-2023	10

Type of data currently in cell: Text / Numeric
1 character(s)

```
PRAGMA foreign_keys = '1',
PRAGMA database_list;
SELECT type,name,sql,tbl_
PRAGMA encoding;
PRAGMA database_list;
SELECT type,name,sql,tbl_
SELECT COUNT(*) FROM "mai
SELECT "_rowid_",* FROM "
```

TEACHER QUESTION TABLE



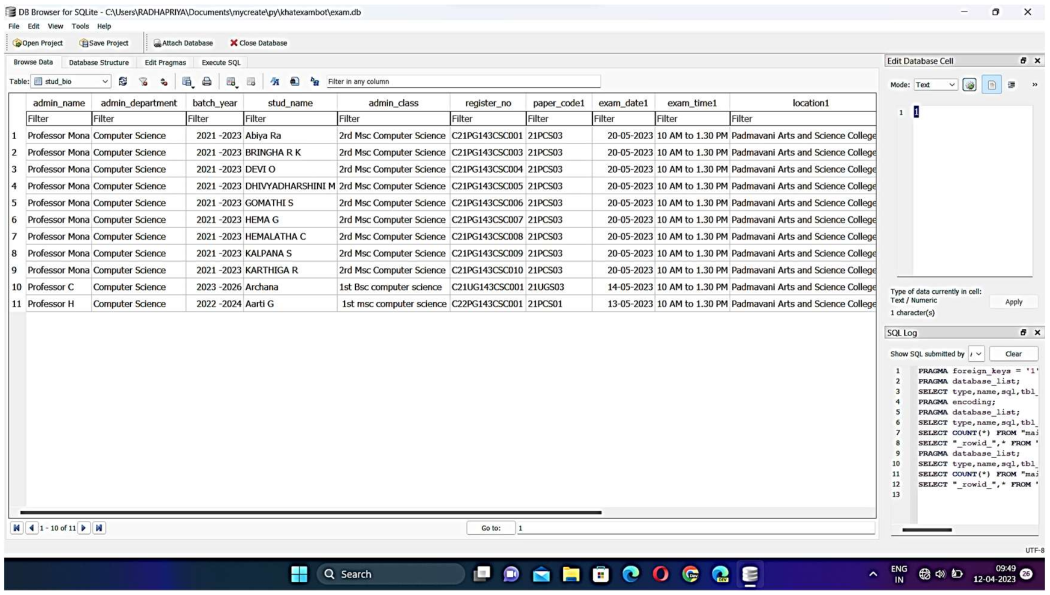
DB Browser for SQLite - C:\Users\RADHAPRIYA\Documents\mycreate\py\khatexambot\exam.db

Table: admin

	admin_name	admin_department	admin_class	batch year
1	Professor Mona	Computer Science	2rd Msc Computer Science | 2rd msc ...	2021 -2023
2	Professor C	Computer Science	1st bsc computer science | 1st bsc cs	2023 -2026
3	Professor H	Computer Science	1st msc computer science| 1st msc cs	2022 -2024

```
PRAGMA foreign_keys = '1'
PRAGMA database_list;
SELECT type,name,sql,tbl_
PRAGMA encoding;
PRAGMA database_list;
SELECT type,name,sql,tbl_
SELECT COUNT(*) FROM "mai
SELECT "_rowid_",* FROM "
```

TEACHER ANSWER TABLE



DB Browser for SQLite - C:\Users\RADHAPRIYA\Documents\mycreate\py\khatexambot\exam.db

Table: stud_bio

	admin_name	admin_department	batch_year	stud_name	admin_class	register_no	paper_code1	exam_date1	exam_time1	location1
1	Professor Mona	Computer Science	2021 -2023	Abiya Ra	2rd Msc Computer Science	C21PG143CSC001	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College
2	Professor Mona	Computer Science	2021 -2023	BRINGHA R K	2rd Msc Computer Science	C21PG143CSC003	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College
3	Professor Mona	Computer Science	2021 -2023	DEVI O	2rd Msc Computer Science	C21PG143CSC004	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College
4	Professor Mona	Computer Science	2021 -2023	DHIVYADHARSHINI M	2rd Msc Computer Science	C21PG143CSC005	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College
5	Professor Mona	Computer Science	2021 -2023	GOMATHI S	2rd Msc Computer Science	C21PG143CSC006	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College
6	Professor Mona	Computer Science	2021 -2023	HEMA G	2rd Msc Computer Science	C21PG143CSC007	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College
7	Professor Mona	Computer Science	2021 -2023	HEMALATHA C	2rd Msc Computer Science	C21PG143CSC008	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College
8	Professor Mona	Computer Science	2021 -2023	KALPANA S	2rd Msc Computer Science	C21PG143CSC009	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College
9	Professor Mona	Computer Science	2021 -2023	KARTHIGA R	2rd Msc Computer Science	C21PG143CSC010	21PCS03	20-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College
10	Professor C	Computer Science	2023 -2026	Archana	1st Bsc computer science	C21UG143CSC001	21UGS03	14-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College
11	Professor H	Computer Science	2022 -2024	Aarti G	1st msc computer science	C22PG143CSC001	21PCS01	13-05-2023	10 AM to 1.30 PM	Padmavani Arts and Science College

```
PRAGMA foreign_keys = '1'
PRAGMA database_list;
SELECT type,name,sql,tbl_
PRAGMA encoding;
PRAGMA database_list;
SELECT type,name,sql,tbl_
SELECT COUNT(*) FROM "mai
SELECT "_rowid_",* FROM "
PRAGMA database_list;
SELECT type,name,sql,tbl_
SELECT COUNT(*) FROM "mai
SELECT "_rowid_",* FROM "
```

10 SYSTEM IMPLEMENTATION AND TESTING

10.1 SYSTEM IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

10.2 SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	:	identified classes of valid input must be accepted.
Invalid Input	:	identified classes of invalid input must be rejected.
Functions	:	identified functions must be exercised.
Output	:	identified classes of application outputs must be exercised.
Systems/Procedures	:	interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

SYSTEM TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document.

It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

UNIT TESTING

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

TEST STRATEGY AND APPROACH

Field testing will be performed manually and functional tests will be written in detail.

TEST OBJECTIVES

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

FEATURES TO BE TESTED

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

TEST RESULTS: All the test cases mentioned above passed successfully. No defects encountered.

ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

TEST RESULTS: All the test cases mentioned above passed successfully. No defects encountered.

11 SOURCE CODE

START PAGE

```
import tkinter as tk

import os

# Create the GUI window

root = tk.Tk()

root.title("PASCW Chatbot")

root.configure(bg="light blue")

root.geometry("600x600")

# Create a canvas for displaying the logo

canvas = tk.Canvas(root, width=500, height=500)

canvas.configure(bg="light blue")

canvas.pack()

# Load the logo image

logo_image = tk.PhotoImage(file="enter.png")

# Display the logo on the canvas

canvas.create_image(250, 250, image=logo_image)

# Define a function to hide the logo and show the next button

def show_next_button():

    canvas.destroy()  # remove the canvas

    next_button.pack()  # show the next button

# Wait for 2 seconds

root.after(2000)

# Create the next button
```

```
def run_main():
    os.system("python tecbot.py")  # execute main.py
next_button = tk.Button(root, text="Next", font=("Arial", 38), command=run_main)
next_button.place(relx=0.7, rely=0.9, anchor=tk.CENTER)
# Start the main event loop
root.mainloop()
```

STUDENT BOT

```
import tkinter as tk
import sqlite3
from nltk.chat.util import Chat, reflections
from prettytable import PrettyTable, prettytable
class ExamSchedulerChatbot:
    def __init__(self, master):
        #windows tab name
        self.master = master
        master.title("Student Chatbot")
        master.configure(bg ="light cyan")
        #connection with sqlite
        self.con = sqlite3.connect('exam_two.db')
        self.c = self.con.cursor()
        #creates GUI frame windows
        self.chat_frame = tk.Frame(master,bg ="light cyan")
        self.chat_frame.pack(side=tk.TOP, padx=10, pady=10)
```

```
#create heading label/title label name

title_label = tk.Label(self.chat_frame, text="Student Exam Schedule Chatbot", bg="light blue", fg="black", font=("Arial", 14))

title_label.pack(side=tk.TOP, fill=tk.X, padx=10, pady=10)

# create chat history text box

self.chat_history = tk.Text(self.chat_frame, height=30, width=99)

self.chat_history.config(state=tk.DISABLED)

self.chat_history.pack(side=tk.LEFT, padx=10, pady=10)

# create scrollbar for chat history

self.chat_scrollbar = tk.Scrollbar(self.chat_frame)

self.chat_scrollbar.pack(side=tk.LEFT, fill=tk.Y)

# configure scrollbar to work with chat history

self.chat_scrollbar.config(command=self.chat_history.yview)

self.chat_history.config(yscrollcommand=self.chat_scrollbar.set)

# entry field of textbox for user

self.chat_input = tk.Entry(self.chat_frame, width=50)

self.chat_input.pack(side=tk.LEFT, padx=10, pady=10)

# submit button

self.chat_submit = tk.Button(self.chat_frame, text="Submit", command=self.submit)

self.chat_submit.pack(side=tk.LEFT, padx=10, pady=10)

# create clear button

self.chat_clear = tk.Button(self.chat_frame, text="Clear", command=self.clear)

self.chat_clear.pack(side=tk.LEFT, padx=10, pady=10)

# Quit button
```

```
        self.chat_quit = tk.Button(self.chat_frame, text="Quit", command=master.quit)
        self.chat_quit.pack(side=tk.RIGHT, padx=10, pady=10)
        #Automatic response by bot
        self.chat("bot: Welcome to Exam Scheduler Chatbot!")
        self.chat("bot: Please enter your Register no:")
        self.context = 'register_no'
 #arrange the message type
    def chat(self, message="", sender="bot"):
        self.chat_history.config(state=tk.NORMAL)
        tag = 'bot' if sender == 'bot' else 'user'
        justify = 'left' if sender == 'bot' else 'right'
        color = 'black' if sender == 'bot' else 'green'
        self.chat_history.tag_configure(tag, justify=justify, foreground=color)
        self.chat_history.insert(tk.END, message + '\n', tag)
        self.chat_history.config(state=tk.DISABLED)
     #get_schedule retrieve the table details from sql /database file
     # this db file has two tables first one named as admin table verify the register no,
department, dob
     # second table will execute the output from stud_bio table by merging with admin
table
    def get_schedule(self, register_no):
      query = """
          SELECT
            exam_schedule.register_no,
            exam_schedule.full_name,
            exam_schedule.student_class,

            exam_schedule.dob_indian,

            exam_schedule.paper_code1,

            exam_schedule.exam_date1,

            exam_schedule.exam_time1,

            exam_schedule.location1,

            exam_schedule.paper_code2,

            exam_schedule.exam_date2,

            exam_schedule.exam_time2,

            exam_schedule.location2

          FROM

            exam_schedule

            INNER    JOIN    students    ON    exam_schedule.register_no    =
students.register_no

          WHERE

            exam_schedule.register_no = ?"""

       self.c.execute(query, (register_no,))

       result = self.c.fetchall()


    # rest of the code

       #if result is present

       if result:

       #create table

         table1 = PrettyTable(['Register no','Student name','Class', 'DOB'])

         table2 = PrettyTable([ 'Paper Code', 'Exam Date', 'Exam Time', 'Location'])
```

```
table3 = PrettyTable([ 'Paper Code 2', 'Exam Date 2', 'Exam Time 2', 'Location 2'])
for row in result:
  table1.add_row([
     row[0],  # department
     row[1],  # Register no
     row[2],  # Class
     row[3]
  ])
  table2.add_row([
     row[4],  # Paper Code
     row[5],  # Exam Date
     row[6],  # Exam Time
     row[7],  # Location
  ])
  table3.add_row([
     row[8],  # Paper Code 2
     row[9],  # Exam Date 2
     row[10], # Exam Time 2
     row[11], # Location 2
  ])
#return table as string
return " Your Exam Schedule:\n  " + str(table1) + "\n\n" + str(table2) + "\n\n" + str(table3)
```

```
   #if result is absent
     else:
        return "Sorry, no exam schedule found for your register no."
      #verify, retrieve, display the data from database file to chatbot windows (output)
   def submit(self):
      user_input = self.chat_input.get() # user's reply
      self.chat_input.delete(0, tk.END)
      response=""
      if self.context == 'register_no':
        self.register_no = user_input # user's reply for Admin name
        response = "Please enter your full name:"
        self.context = 'full_name'
      elif self.context == 'full_name':
        self.full_name = user_input # user's reply for department
        response = "Please enter your class:"
        self.context = 'student_class'
      elif self.context == 'student_class':
        self.admin_class = user_input # user's reply for class
        response = "Please enter your date of birth (dd/mm/yyyy):"
        self.context = 'dob_indian'
      elif self.context == 'dob_indian':
        self.batch_year = user_input # user's reply for date of birth
        response = self.get_schedule(self.register_no) # gets value using admin_name
as main foreign key in sql
```

```
            self.context = 'register_no'

        self.chat("You: " + user_input, sender='user') # User response
        self.chat("bot: " + response,sender='bot') # bot response
        self.chat_input.delete(0, tk.END) # deletes the text in entry textbox field

    def clear(self):
        self.chat_input.delete(0, tk.END)
        self.chat_history.config(state=tk.NORMAL)
        self.chat_history.delete('2.0', tk.END) # only delete user's input, leave bot's messages
        self.chat_history.config(state=tk.DISABLED)
        self.context = 'register_no'
    # check if there are any bot messages in chat history
        chat_lines = self.chat_history.get('2.0', tk.END).split('\n')
        bot_messages = [line for line in chat_lines if line.startswith('bot:')]
        if len(bot_messages) == 0:
            self.chat(" Please enter your Register no:")
        else:
            self.chat(bot_messages[0])
            self.chat(bot_messages[1]
    def run(self):
        self.master.mainloop()

root = tk.Tk()
```

```
chatbot = ExamSchedulerChatbot(root) #end of gui windows

chatbot.run()
```

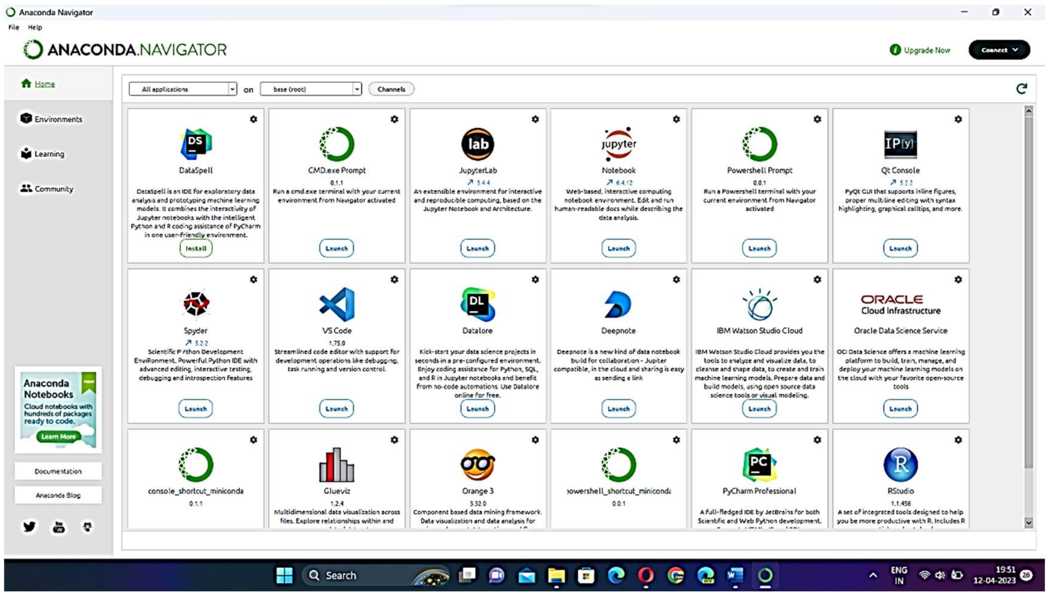
TEACHER BOT

```
import tkinter as tk

import sqlite3

from nltk.chat.util import Chat, reflections

from prettytable import PrettyTable, prettytable

class Teacherbot:

    def __init__(self, master):

        #windows tab name

        self.master = master

        master.title("Teacher Chatbot")

        master.configure(bg ="light cyan")

        #connection with sqlite

        self.con = sqlite3.connect('exam.db')

        self.c = self.con.cursor()

        #creates GUI frame windows

        self.chat_frame = tk.Frame(master,bg ="light cyan")

        self.chat_frame.pack(side=tk.TOP, padx=10, pady=10)

        #create heading label/title label name

        title_label = tk.Label(self.chat_frame, text="Teacher Exam Schedule Chatbot", 
bg="light blue", fg="black", font=("Arial", 14))

        title_label.pack(side=tk.TOP, fill=tk.X, padx=10, pady=10)


        # create chat history text box
        self.chat_history = tk.Text(self.chat_frame, height=30, width=99)
        self.chat_history.config(state=tk.DISABLED)
        self.chat_history.pack(side=tk.LEFT, padx=10, pady=10)

        # create scrollbar for chat history
        self.chat_scrollbar = tk.Scrollbar(self.chat_frame)
        self.chat_scrollbar.pack(side=tk.LEFT, fill=tk.Y)

        # configure scrollbar to work with chat history
        self.chat_scrollbar.config(command=self.chat_history.yview)
        self.chat_history.config(yscrollcommand=self.chat_scrollbar.set)

        # entry field of textbox for user
        self.chat_input = tk.Entry(self.chat_frame, width=50)
        self.chat_input.pack(side=tk.LEFT, padx=10, pady=10)

        # submit button
        self.chat_submit = tk.Button(self.chat_frame, text="Submit",
command=self.submit)
        self.chat_submit.pack(side=tk.LEFT, padx=10, pady=10)
        # create clear button
        self.chat_clear = tk.Button(self.chat_frame, text="Clear", command=self.clear)
        self.chat_clear.pack(side=tk.LEFT, padx=10, pady=10)

        # Quit button
        self.chat_quit = tk.Button(self.chat_frame, text="Quit", command=master.quit)

        self.chat_quit.pack(side=tk.RIGHT, padx=10, pady=10)

        #Automatic response by bot

        self.chat("bot: Welcome to Exam Scheduler Chatbot!")

        self.chat("bot: Please enter your Admin name:")

        self.context = 'admin_name'

    #arrange the message type

    def chat(self, message="", sender="bot"):

        self.chat_history.config(state=tk.NORMAL)

        tag = 'bot' if sender == 'bot' else 'user'

        justify = 'left' if sender == 'bot' else 'right'

        color = 'black' if sender == 'bot' else 'green'

        self.chat_history.tag_configure(tag, justify=justify, foreground=color)

        self.chat_history.insert(tk.END, message + '\n', tag)

        self.chat_history.config(state=tk.DISABLED)

     #get_schedule retrieve the table details from sql /database file

     # this db file has two tables first one named as admin table verify the register no,
department, dob

     # second table will execute the output from stud_bio table by merging with admin
table

    def get_schedule(self, admin_name):

      query = """

            SELECT

              stud_bio.stud_name,

              stud_bio.admin_class,
              stud_bio.register_no,
              stud_bio.batch_year,
              stud_bio.paper_code1,
              stud_bio.exam_date1,
              stud_bio.exam_time1,
              stud_bio.location1,
              stud_bio.paper_code2,
              stud_bio.exam_date2,
              stud_bio.exam_time2,
              stud_bio.location2
          FROM
              stud_bio
              INNER JOIN admin ON stud_bio.admin_name = admin.admin_name
          WHERE
              stud_bio.admin_name = ?"""
      self.c.execute(query, (admin_name,))
      result = self.c.fetchall()
    # rest of the code
        #if result is present
      if result:
        #create table
         table1 = PrettyTable(['Student name','Register no','Class'])
         table2 = PrettyTable([ 'Paper Code', 'Exam Date', 'Exam Time', 'Location'])
         table3 = PrettyTable([ 'Paper Code 2', 'Exam Date 2', 'Exam Time 2', 'Location
2'])
```

```
        for row in result:
          table1.add_row([
              row[0],  # department
              row[1],  # Register no
              row[2],  # Class
          ])
          table2.add_row([
              row[4],  # Paper Code
              row[5],  # Exam Date
              row[6],  # Exam Time
              row[7],  # Location
          ])
          table3.add_row([
              row[8],  # Paper Code 2
              row[9],  # Exam Date 2
              row[10], # Exam Time 2
              row[11], # Location 2
          ])
        #return table as string
        return " Your Class Exam Schedule:\n  " + str(table1) + "\n\n" + str(table2) +
"\n\n" + str(table3)
    #if result is absent
      else:
          return "Sorry, no exam schedule found for your Admin name."
       #verify, retrieve, display the data from database file to chatbot windows (output)
    def submit(self):
        user_input = self.chat_input.get() # user's reply
        self.chat_input.delete(0, tk.END)
        if self.context == 'admin_name':
            self.admin_name = user_input # user's reply for Admin name
            response = "Please enter your department:"
            self.context = 'admin_department'
        elif self.context == 'admin_department':
            self.admin_department = user_input # user's reply for department
            response = "Please enter your class:"
            self.context = 'class'
        elif self.context == 'class':
            self.admin_class = user_input # user's reply for class
            response = "Please enter your batch year:"
            self.context = 'batch_year'
        elif self.context == 'batch_year':
            self.batch_year = user_input # user's reply for date of birth
            response = self.get_schedule(self.admin_name) # gets value using admin_name as main foreign key in sql
            self.context = 'admin_name'
        self.chat("You: " + user_input, sender='user') # User response
        self.chat("bot: " + response,sender='bot') # bot response
        self.chat_input.delete(0, tk.END) # deletes the text in entry textbox field


    def clear(self):
        self.chat_input.delete(0, tk.END)

        self.chat_history.config(state=tk.NORMAL)

        self.chat_history.delete('2.0', tk.END) # only delete user's input, leave bot's
messages

        self.chat_history.config(state=tk.DISABLED)

        self.context = 'admin_name'

    # check if there are any bot messages in chat history

        chat_lines = self.chat_history.get('2.0', tk.END).split('\n')

        bot_messages = [line for line in chat_lines if line.startswith('bot:')]

        if len(bot_messages) == 0:

          self.chat(" Please enter your Admin name:")

        else:

          self.chat(bot_messages[0])

          self.chat(bot_messages[1])

    def run(self):

        self.master.mainloop()

root = tk.Tk()

chatbot = Teacherbot(root) #end of gui windows

chatbot.run()
```

12 SCREENSHOT

ANACONDA STARTS



DataSpell

DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.

CMD.exe Prompt 0.1.1

Run a cmd.exe terminal with your current environment from Navigator activated

JupyterLab 3.4.4

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook 6.4.12

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Powershell Prompt 0.0.1

Run a Powershell terminal with your current environment from Navigator activated

Qt Console 5.2.2

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder 5.2.2

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

VS Code 1.75.0

Streamlined code editor with support for development operations like debugging, task running and version control.

Datalore

Kick-start your data science projects in seconds in a pre-configured environment. Enjoy coding assistance for Python, SQL, and R in Jupyter notebooks and benefit from no-code automations. Use Datalore online for free.

Deepnote

Deepnote is a new kind of data notebook build for collaboration - Jupiter compatible, in the cloud and sharing is easy as sending a link

IBM Watson Studio Cloud

IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

ORACLE Cloud Infrastructure
Oracle Data Science Service

OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source tools

console_shortcut_miniconda 0.1.1

Glueviz 1.2.4

Multidimensional data visualization across files. Explore relationships within and

Orange 3 3.32.0

Component based data mining framework. Data visualization and data analysis for

›owershell_shortcut_minicond‹ 0.0.1

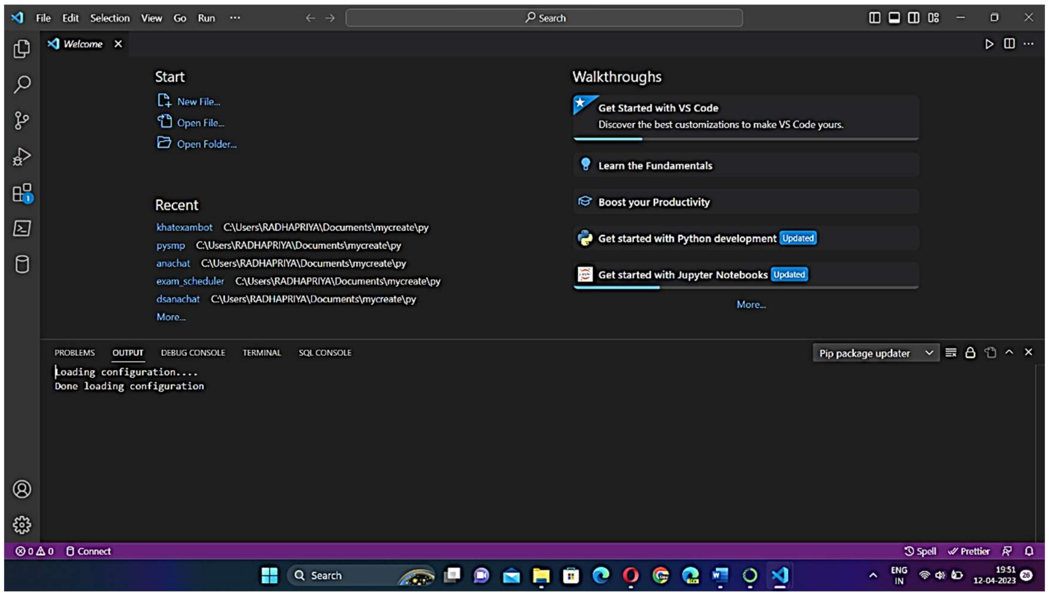
PyCharm Professional

A full-fledged IDE by JetBrains for both Scientific and Web Python development.

RStudio 1.1.456

A set of integrated tools designed to help you be more productive with R. Includes R

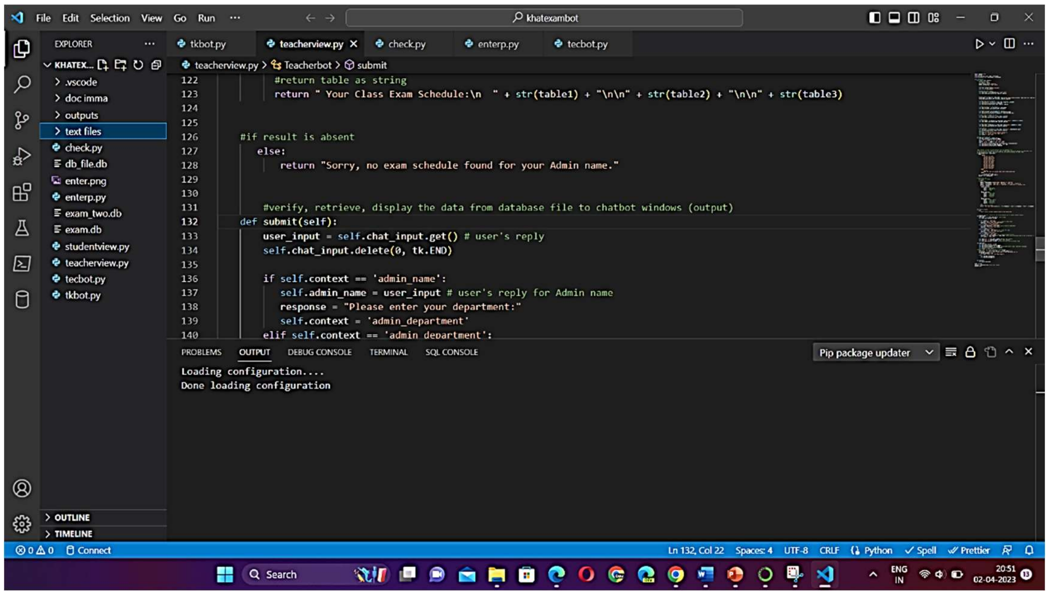
VS CODE STARTS



khatexambot C:\Users\RADHAPRIYA\Documents\mycreate\py
pysmp C:\Users\RADHAPRIYA\Documents\mycreate\py
anachat C:\Users\RADHAPRIYA\Documents\mycreate\py
exam_scheduler C:\Users\RADHAPRIYA\Documents\mycreate\py
dsanachat C:\Users\RADHAPRIYA\Documents\mycreate\py
More...

```
Loading configuration....
Done loading configuration
```

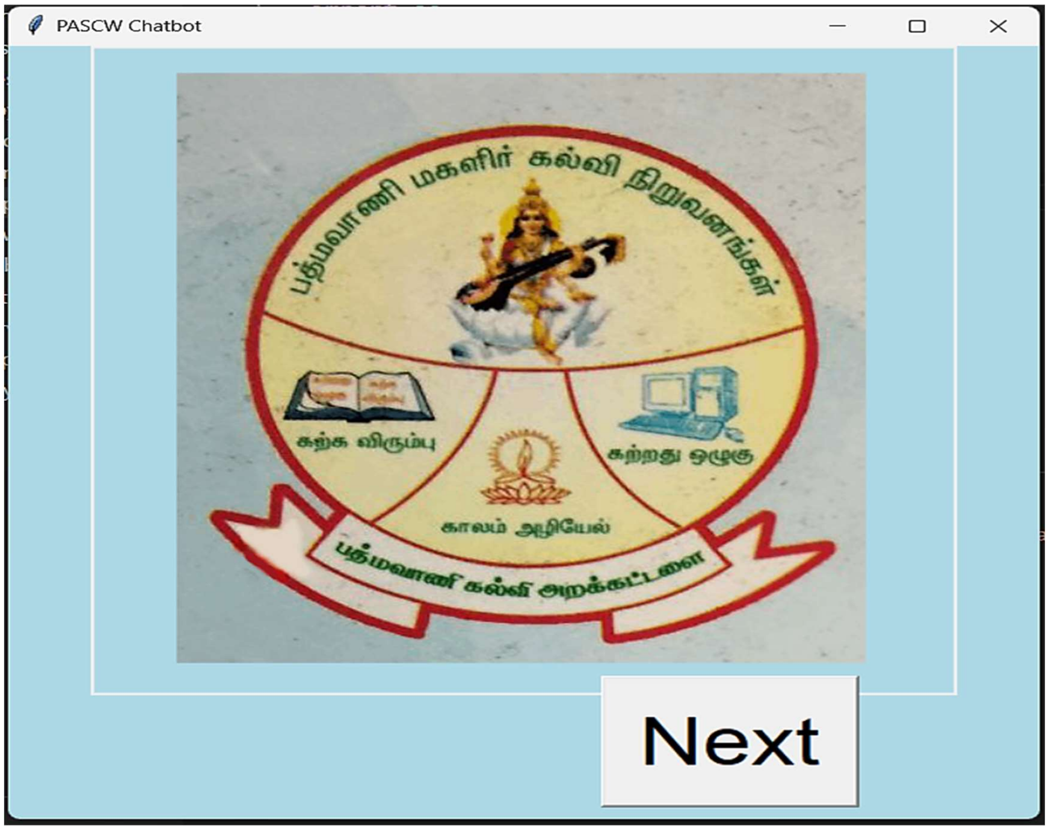
SOURCE CODE PAGE



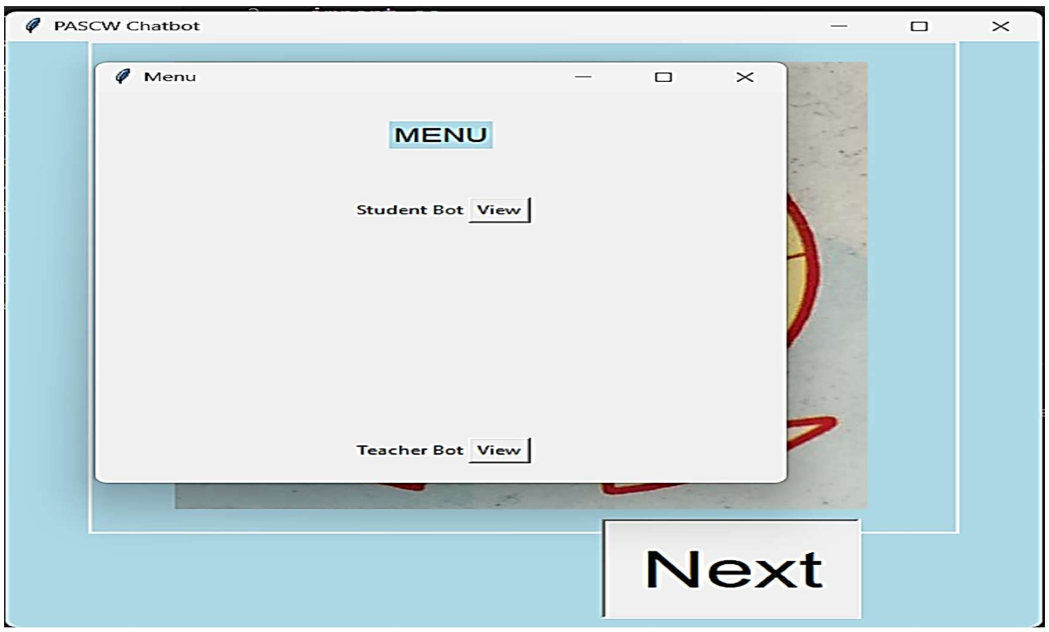
```
            #return table as string
            return " Your Class Exam Schedule:\n  " + str(table1) + "\n\n" + str(table2) + "\n\n" + str(table3)


    #if result is absent
       else:
           return "Sorry, no exam schedule found for your Admin name."


        #verify, retrieve, display the data from database file to chatbot windows (output)
    def submit(self):
        user_input = self.chat_input.get() # user's reply
        self.chat_input.delete(0, tk.END)

        if self.context == 'admin_name':
           self.admin_name = user_input # user's reply for Admin name
           response = "Please enter your department:"
           self.context = 'admin_department'
        elif self.context == 'admin_department':
```

```
Loading configuration....
Done loading configuration
```

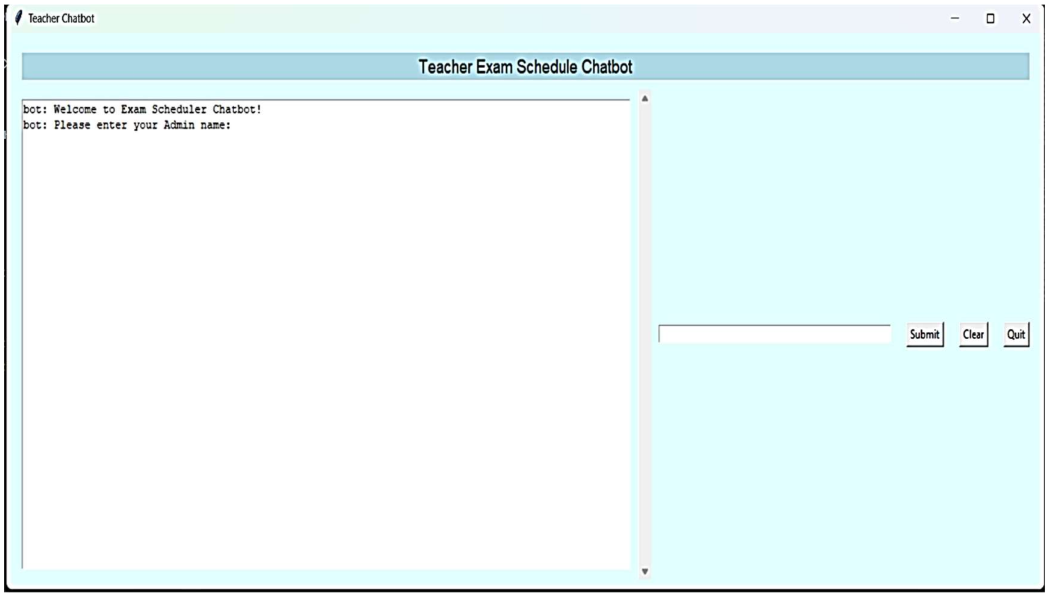
HOME PAGE



MENU

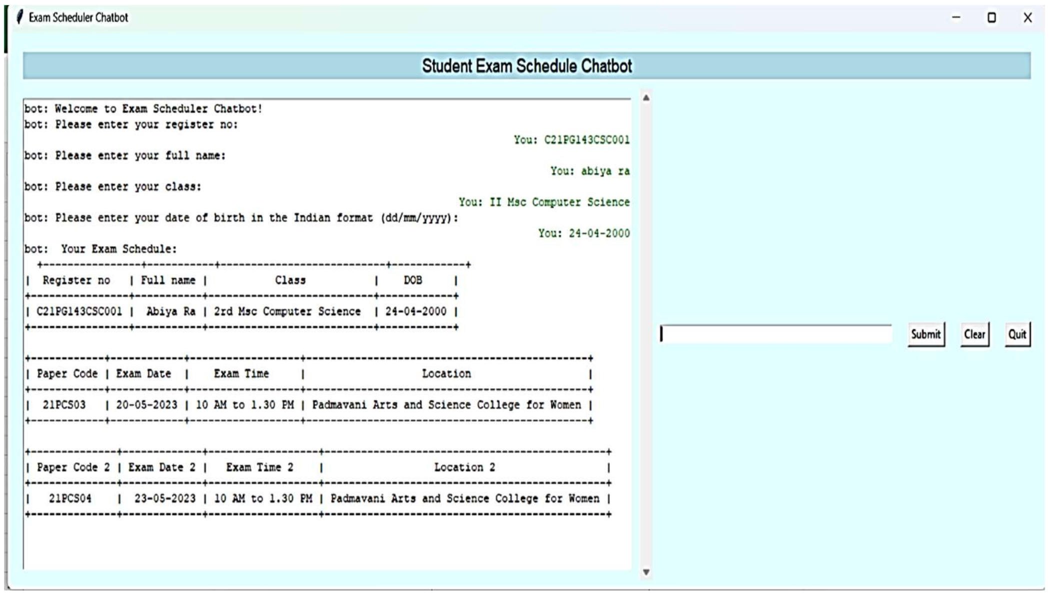


START



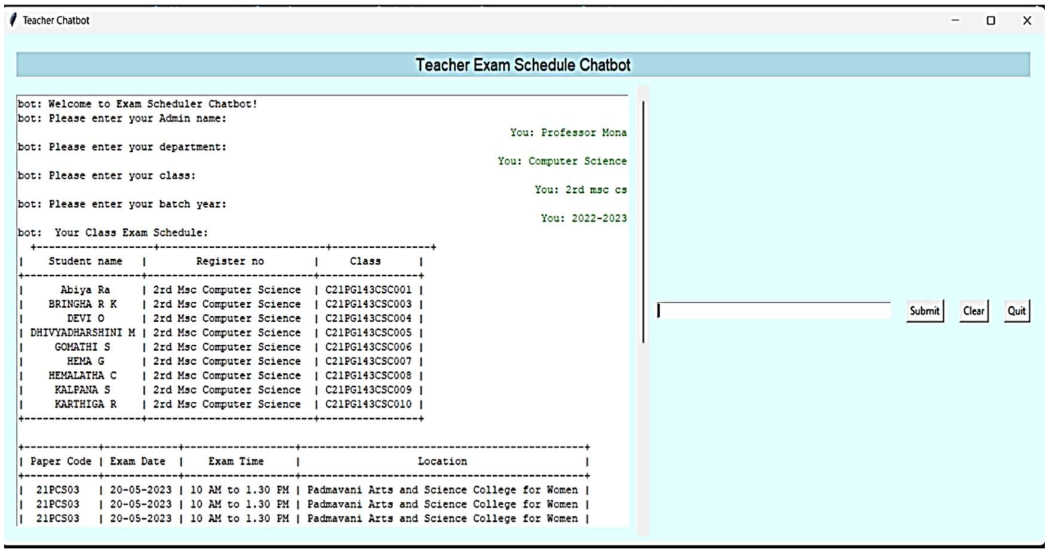
```
bot: Welcome to Exam Scheduler Chatbot!
bot: Please enter your Admin name:
```

STUDENT BOT OUTPUT



```
bot: Welcome to Exam Scheduler Chatbot!
bot: Please enter your register no:
                                                              You: C21PG143CSC001
bot: Please enter your full name:
                                                                    You: abiya ra
bot: Please enter your class:
                                                   You: II Msc Computer Science
bot: Please enter your date of birth in the Indian format (dd/mm/yyyy):
                                                                  You: 24-04-2000
bot:  Your Exam Schedule:
  +----------------+-----------+--------------------------+------------+
| Register no    | Full name |          Class           |    DOB     |
+----------------+-----------+--------------------------+------------+
| C21PG143CSC001 |  Abiya Ra | 2rd Msc Computer Science  | 24-04-2000 |
+----------------+-----------+--------------------------+------------+

+------------+------------+-------------------+---------------------------------------------+
| Paper Code | Exam Date  |    Exam Time      |                  Location                   |
+------------+------------+-------------------+---------------------------------------------+
|  21PCS03   | 20-05-2023 | 10 AM to 1.30 PM | Padmavani Arts and Science College for Women |
+------------+------------+-------------------+---------------------------------------------+

+--------------+--------------+-------------------+---------------------------------------------+
| Paper Code 2 | Exam Date 2 |   Exam Time 2     |                  Location 2                  |
+--------------+--------------+-------------------+---------------------------------------------+
|   21PCS04    |  23-05-2023 | 10 AM to 1.30 PM | Padmavani Arts and Science College for Women |
+--------------+--------------+-------------------+---------------------------------------------+
```

TEACHER BOT OUTPUT



```
bot: Welcome to Exam Scheduler Chatbot!
bot: Please enter your Admin name:
                                                            You: Professor Mona
bot: Please enter your department:
                                                          You: Computer Science
bot: Please enter your class:
                                                                You: 2rd msc cs
bot: Please enter your batch year:
                                                                 You: 2022-2023
bot:  Your Class Exam Schedule:
  +-------------------+--------------------------+----------------+
|    Student name    |       Register no        |     Class      |
+-------------------+--------------------------+----------------+
|      Abiya Ra      | 2rd Msc Computer Science | C21PG143CSC001 |
|    BRINGHA R K     | 2rd Msc Computer Science | C21PG143CSC003 |
|       DEVI O       | 2rd Msc Computer Science | C21PG143CSC004 |
| DHIVYADHARSHINI M | 2rd Msc Computer Science | C21PG143CSC005 |
|     GOMATHI S      | 2rd Msc Computer Science | C21PG143CSC006 |
|       HEMA G       | 2rd Msc Computer Science | C21PG143CSC007 |
|    HEMALATHA C     | 2rd Msc Computer Science | C21PG143CSC008 |
|     KALPANA S      | 2rd Msc Computer Science | C21PG143CSC009 |
|     KARTHIGA R     | 2rd Msc Computer Science | C21PG143CSC010 |
+-------------------+--------------------------+----------------+

+------------+------------+-------------------+---------------------------------------------+
| Paper Code | Exam Date  |    Exam Time      |                  Location                   |
+------------+------------+-------------------+---------------------------------------------+
|  21PCS03   | 20-05-2023 | 10 AM to 1.30 PM | Padmavani Arts and Science College for Women |
|  21PCS03   | 20-05-2023 | 10 AM to 1.30 PM | Padmavani Arts and Science College for Women |
|  21PCS03   | 20-05-2023 | 10 AM to 1.30 PM | Padmavani Arts and Science College for Women |
```

13. CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION

The suggested model has proven its effectiveness in terms of creating a conflict-free exam schedule allowing students to find their exams in a relaxed environment and enabling the faculty's study and examination office staff to announce an exam schedule that meets the criteria and standard established for the open semester system. Although we designed this application for the faculty, believe it can be applied to other faculties, as well as other institutions that use an open semester as a study system. We plan to further improve the application so that it will be able to schedule the exam's monitor timetable, and the exam's halls where exams will be held, Examination scheduler will be an integrated system.

In future work, all system interfaces of the chatbot should also have a built-in AI system and, for account security, a login system. In addition, applying miscellaneous logic for the automatic scheduler to consider allocating the proctors to the users could also be beneficial since this is one of the optional suggestions of the business clients. Significantly, the researchers assert that these enhancements will provide high satisfaction and positive innovation to the mentioned college department.

BIBILOGRAPHY

[1] A. F.Abuhamdah, “A Dynamic System for Real-World University Examination Timetabling Problem Using AAC approach Case Study: Taibah University-CCSE,” International Journal of Applied Engineering Research (IJAER), vol. 13, no. 8, p. 12, 2020.

[2] M. A. Moreb,“Final Exam Scheduling Timetable a Case Study,” p. 16, May 2020.

[3] A. k. Mandal and M. N. M. Kahar, “Solving Examination Timetabling Problem using Partial Exam Assignment with Great Deluge Algorithm,” IEEE, 2021.

[4] Z. Bratković, T. Herman, V. Omrčen, M. Čupić and D. Jakobović, “University course timetabling with genetic algorithm: A laboratory excercises case study,” in European Conference on Evolutionary Computation in Combinatorial Optimization, Springer, 2022, pp. 240- -251.

[5] M. Ayob, A. R. Hamdan, S. Abdullah, Z. Othman, M. Z. Ahmad, K. A. Razak, R. Tan, N. Baharom, H. A. Ghafar, R. M. Dali and N. R. Sabar, “Intelligent Examination Timetabling Software,” science direct, vol. 18, pp. 600-608, 2019.

[6] R. O. Vrielink, E. A. Jansen2, E. W. Hans and J. v. Hillegersberg, “Practices in timetabling in higher education institutions: a systematic review,” Annals of operations research, vol. 275, no. 1, pp. 145--160, 27 10 2020.

[7] M. Čupić, M. Golub and D. Jakobovic, “Exam Schedule Using Genetic Algorithm,” IEEE, 2020.

[8] M. DENER and M. H. CALP, “SOLVING THE EXAM SCHEDULING PROBLEMS IN CENTRAL EXAMS WITH GENETIC ALGORITHMS,” Mugla Journal of Science and Technology, p. 14, 2020.

[9] T. Müller, “Real-life examination timetabling,” Journal of Scheduling, August 2020.

[10] S. Erdős and B. Kővári, “GENETIC ALGORITHM BASED SOLUTION FOR FINAL EXAM SCHEDULING,” MultiScience - XXXIII. microCAD International Multidisciplinary Scientific Conference, 23-24 May 2020.

[11] A. Yasin, G. M. Jaradat and R. Daud, "Secondary School Examination Timetabling using Genetic Algorithm," Lampiran Tabel, vol. 3, November 2021.

[12] A. Jain, G. S. C. Aiyer, H. Goel and R. Bhandari, "A Literature Review on Timetable generation algorithms based on Genetic Algorithm and Heuristic approach," International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, no. 4, p. 5, April 2019.

[13] A. Schaerf, "A Survey of Automated Timetabling," Artificial intelligence review, 2018.

[14] J. Xu, "Improved Genetic Algorithm to Solve the Scheduling Problem of College English Courses," Complexity, vol. 2021, p. 11, 2019.

[15] R. Qu, E. k. Burke, B. McCollum, L. T. Merlot and S. Y. Lee, "A survey of search methodologies and automated system development for examination timetabling," Journal of scheduling, vol. 12, no. 1, pp. 55--89, 2019.

[16] A. Lawrynowicz, "Genetic algorithms for solving scheduling problems in manufacturing systems," Foundations of Management, vol. 3, no. 2, pp. 7--26, 2019.

[17] J. Zhang, Z.-h. Zhan, Y. Lin, N. Chen, Y.-j. Gong, J.-h. Zhong, H. S. Chung, Y. Li and Y.-h. Shi, "Evolutionary Computation Meets Machine Learning: A Survey," IEEE Computational Intelligence Magazine, vol. 6, no. 4, pp. 68-75, 2018.

[18] A. A. Alli, M. S ., F. Sanz L., A. A. O. and E. A. R., "Genetic algorithm and tabu search memory with course sandwiching (GATS_CS) for university examination timetabling," Intelligent Automation and Soft Computing, vol. 26, pp. 385--396, 2020.

[19] M. A. Mohammed, M. Ghani, O. I. Obaid, S. A. Mostafa, M. S. Ahmad, M. S. Ibrahim and M. Burhanuddin, "A Review of Genetic Algorithm Application in Examination Timetabling Problem," Journal of Engineering and Applied Sciences, vol. 12, no. 20, pp. 5166--5181, 2021.

[20] J. J. Moreira, "A System of Automatic Construction of Exam Timetable Using Genetic Algorithms," Tékhne-Revista de Estudos Politécnicos, vol. VI, pp. 319--336, 2018.