**PRESTIGE CONVENT SCHOOL**

POCKET-B-5, SECTOR-8, ROHINI, NEW DELHI-110085

**COMPUTER SCIENCE**

CLASS XII-A

**PROJECT FILE**

**“PERSONAL FINANCE MANEGER”**

****

**AISSCE 2024-25 EXAMINATION**

**As a part of the Computer Science Course (083)**

**SUBMITTED BY- SUBMITTED TO-**

NIPUN GUPTA MS. AMBIKA SHARMA

P.G.T (Comp. Sc.)

**ROLL NO:**

**CERTIFICATE**

This is to certify that the project entitled “PERSONAL FINANCE MANEGER” is a bona fide work done by “NIPUN GUPTA” of class XII-A, session “2024-25”, in partial fulfilment of CBSE’s AISSCE examination. This report has not been submitted for any other examination and does not form a part of any other course undergone by any candidate.

------------------- -------------------- --------------------

Sign. of student Sign. of Teacher Sign. Of Principal

Nipun Gupta Ms. Ambika Sharma Inderjit Kaur Batra

Roll No: \_\_\_\_\_\_\_

Place: ROHINI, DELHI

Date:\_\_\_\_\_\_\_\_\_

**AKNOWLEDGEMENT**

I undertook this project, as a part of my Class XII, Computer Science course. I had tried to apply my best of knowledge and experience gained during the studies and class work experience.

However, a developing software system is generally quite complex and time consuming process. Moreover, the developer always feel the need, the help and good wishes of the people near, who have considerable experience and idea.

I would like to extend my sincere thanks and gratitude to my teacher, Ms. AMBIKA SHARMA

I would also like to extend my sincere thanks and gratitude to my principal, Ms. INDERJEET KAUR BATRA.

I would like to take the opportunity to extend my sincere thanks and gratitude to my parents.

For being a source of inspiration and providing time and freedom to develop this software project.

NIPUN GUPTA

XII-A

**Index**

|  |  |
| --- | --- |
| **S.No.** | **TOPIC** |
| 1. | AIM |
| 2. | INTRODUCTION |
| 3. | OBJECTIVES |
| 4. | REASONS |
| 5. | GOALS |
| 6. | ADVANTAGES |
| 7. | DISADVANTAGES |
| 8. | HARDWARE/SOFTWARE REQUIRED |
| 9. | PYTHON |
| 10. | DATABASE |
| 11. | MODULES USED |
| 12. | CODING |
| 13. | FUTURE SCOPE |
| 14. | CONCLUSION |
| 15. | REFERENCES |

**AIM**

**Aim of the Project**

The aim of the Personal Finance Manager project is to develop an application that helps users manage their personal finances efficiently. The application enables users to track their expenses, manage their budgets, and generate financial reports, thereby providing a comprehensive tool for financial management and planning.

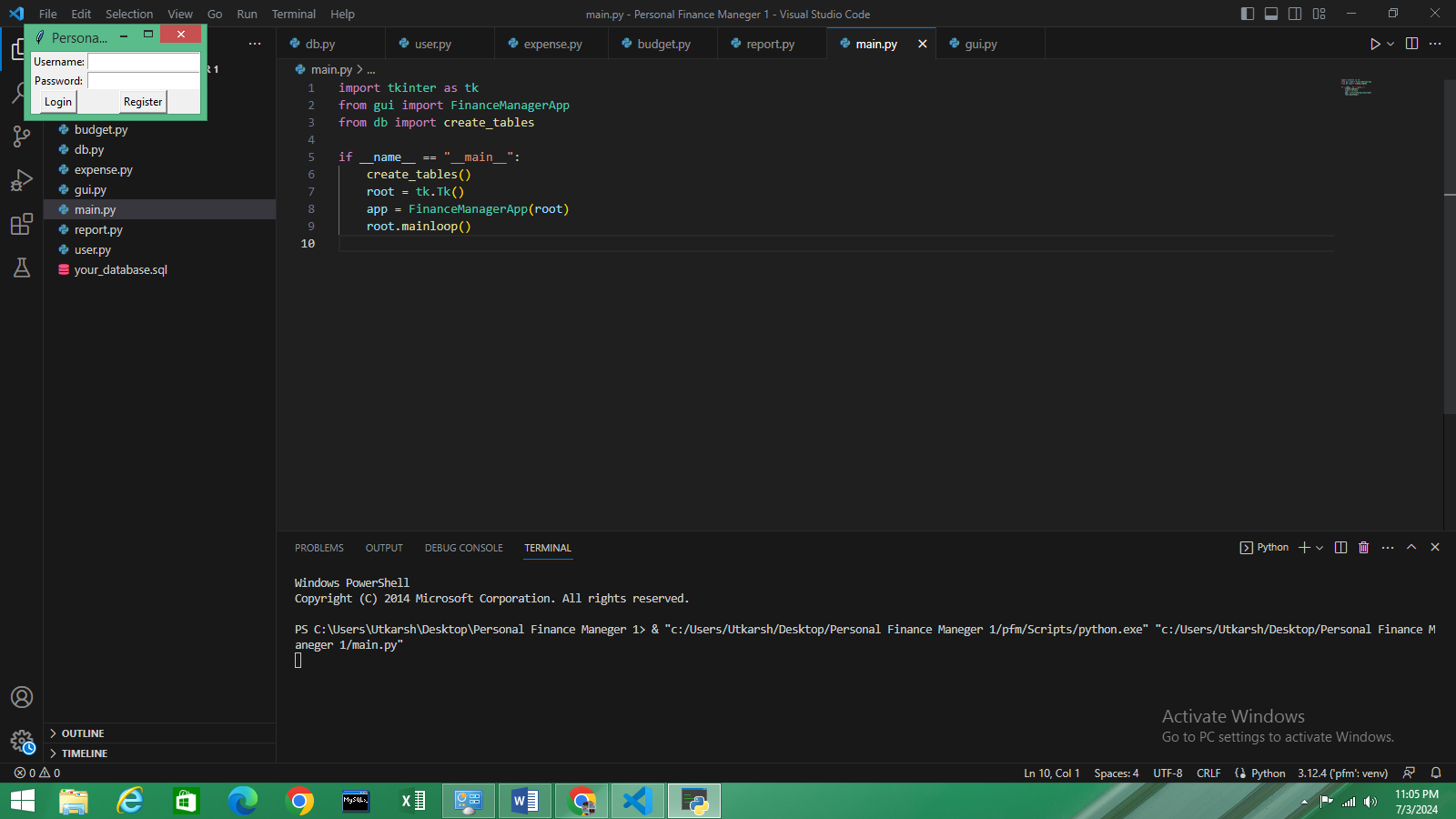
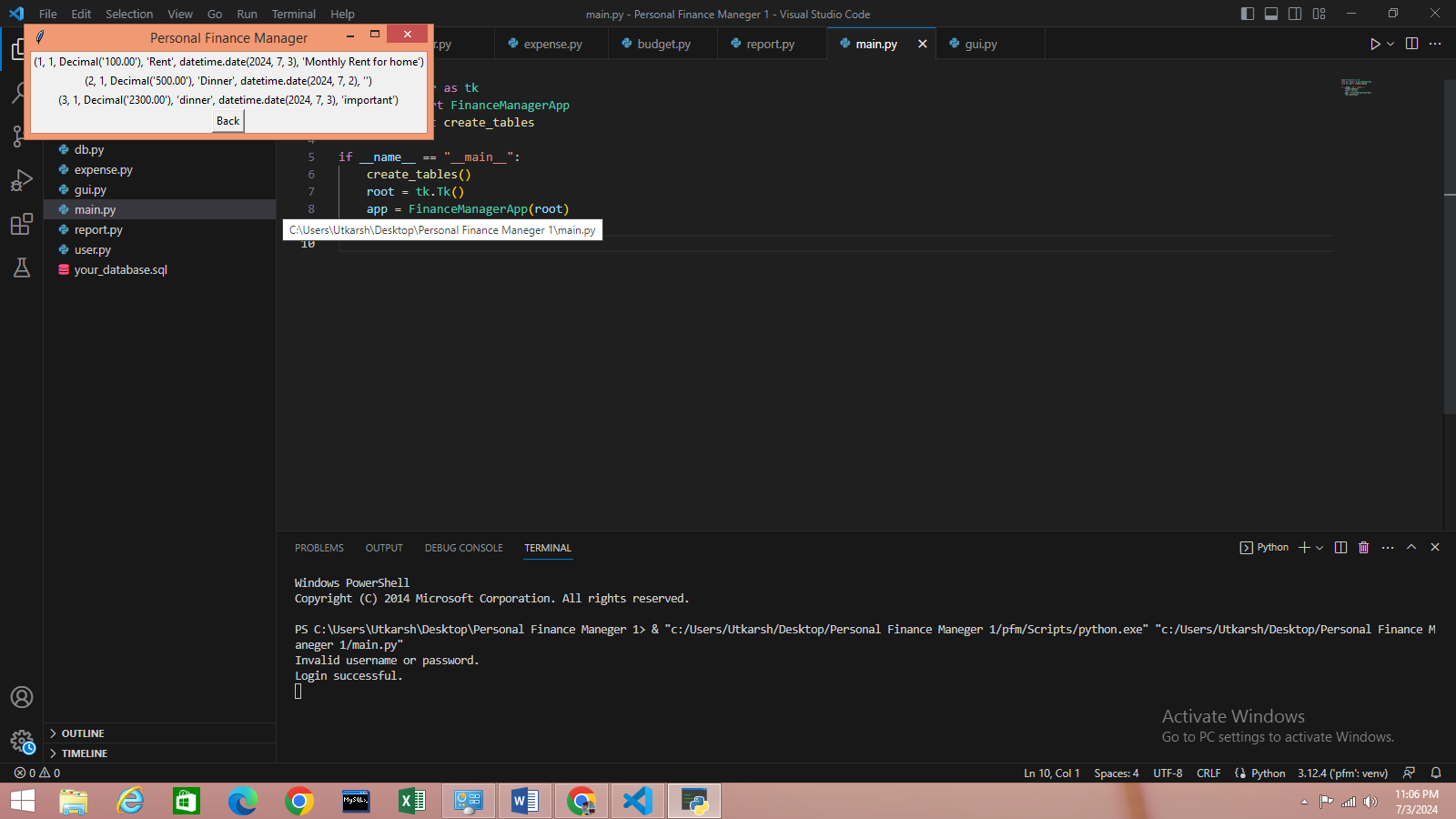
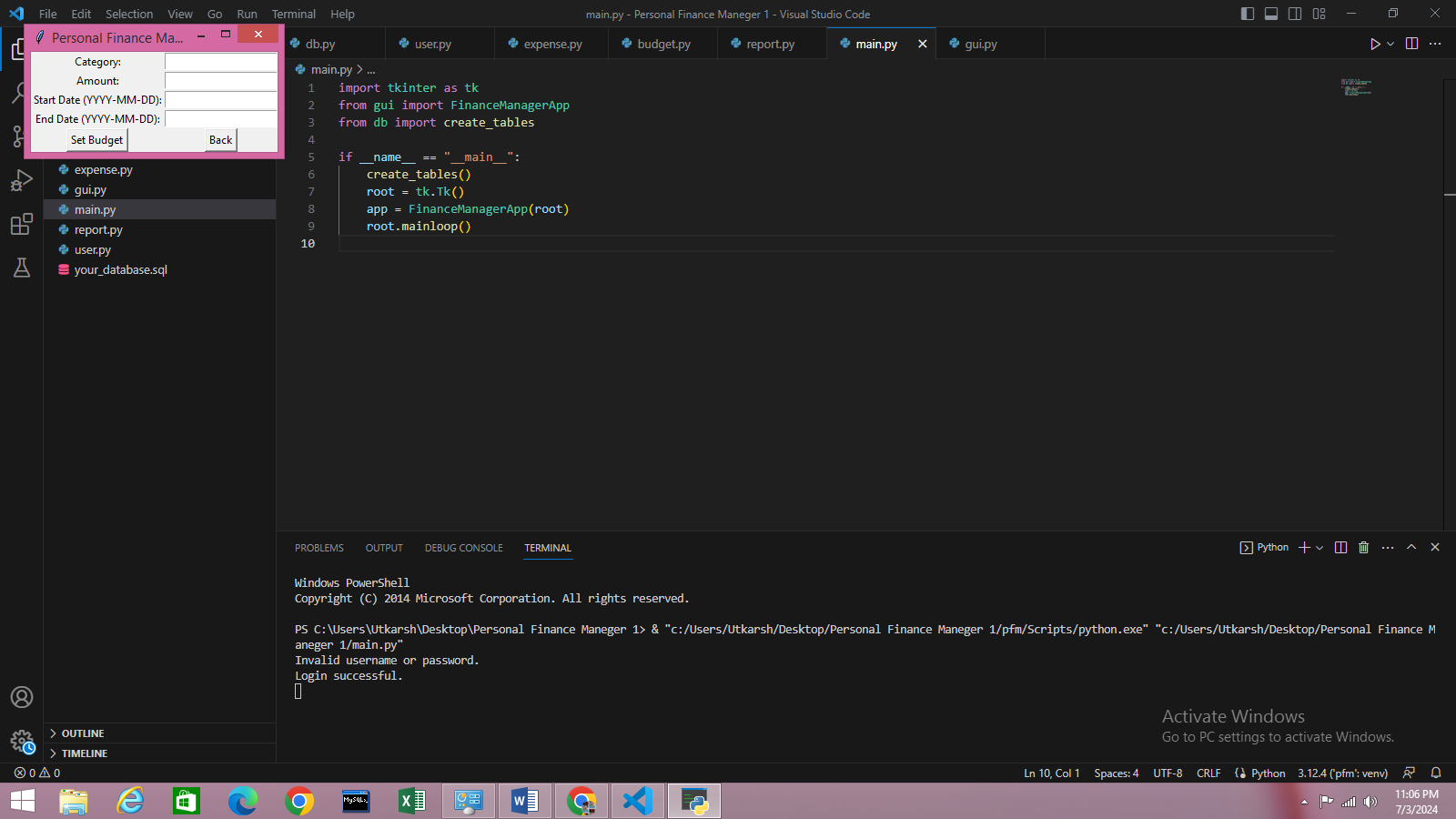
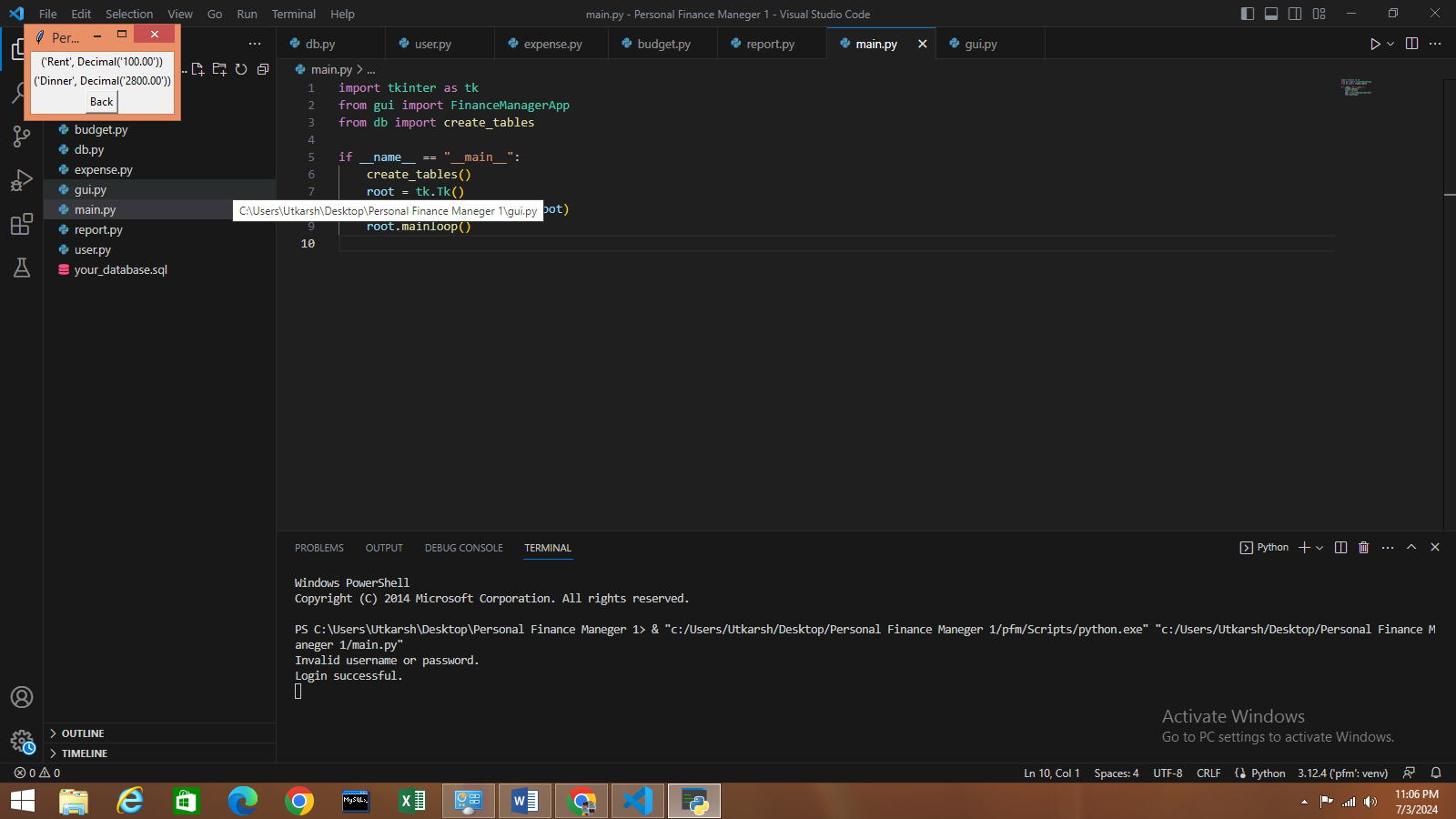
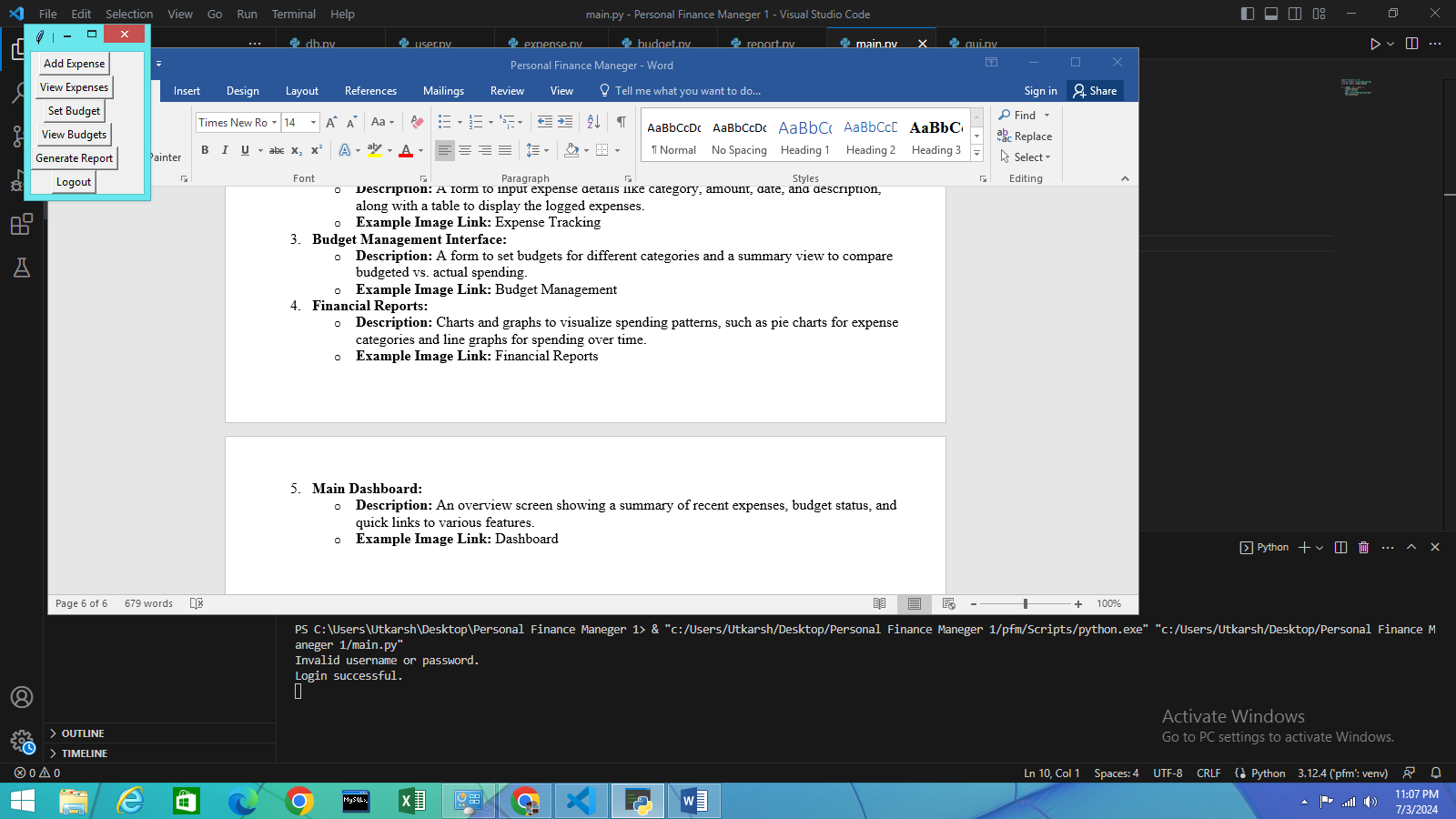
**Detailed Description**

**Key Objectives:**

1. **Expense Tracking:**
   * Allow users to log their daily expenses categorized by type (e.g., food, transportation, entertainment).
   * Enable users to view their expenses over different time periods (daily, weekly, monthly).
2. **Budget Management:**
   * Provide functionality for users to set budgets for different expense categories.
   * Track the actual spending against the set budgets and notify users when they are nearing or exceeding their budgets.
3. **Financial Reporting:**
   * Generate reports to provide users with insights into their spending habits.
   * Visualize data through charts and graphs for better understanding and analysis.
4. **User Authentication:**
   * Ensure secure user registration and login functionalities to protect user data.
5. **User-Friendly Interface:**
   * Develop a graphical user interface (GUI) using Tkinter for ease of use.
   * Make the application intuitive and easy to navigate.

**Visual Representation**

As I cannot directly provide images here, I can describe what each part of the application might look like and suggest how you can create or find appropriate images.

1. **User Registration and Login Screen:**
   * **Description:** A simple form with fields for username and password, and buttons for "Register" and "Login".
2. **Expense Tracking Interface:**
   * **Description:** A form to input expense details like category, amount, date, and description, along with a table to display the logged expenses.
3. **Budget Management Interface:**
   * **Description:** A form to set budgets for different categories and a summary view to compare budgeted vs. actual spending.
4. **Financial Reports:**
   * **Description:** Charts and graphs to visualize spending patterns, such as pie charts for expense categories and line graphs for spending over time.
5. **Main Dashboard:**
   * **Description:** An overview screen showing a summary of recent expenses, budget status, and quick links to various features.

# Personal Finance Manager

## Introduction

Managing personal finances can be a challenging task, especially in today's fast-paced world where expenses can quickly spiral out of control. To address this challenge, we present the **Personal Finance Manager**, an application designed to simplify the process of managing personal finances. This project aims to provide users with a comprehensive tool to track their expenses, manage their budgets, and gain valuable insights into their financial habits through detailed reports.

The Personal Finance Manager is developed using Python, leveraging its robust libraries and tools to create an efficient and user-friendly application. MySQL is used as the database management system to ensure secure and reliable storage of user data, while Tkinter is utilized to create an intuitive graphical user interface (GUI).

### Key Features

1. **Expense Tracking**: Users can log their daily expenses, categorize them, and view detailed records over various time periods.
2. **Budget Management**: The application allows users to set budgets for different categories and track their spending against these budgets.
3. **Financial Reporting**: Users can generate reports to analyse their spending habits and visualize their financial data through charts and graphs.
4. **User Authentication**: Secure user registration and login functionalities protect user data and ensure privacy.
5. **User-Friendly Interface**: The GUI is designed to be intuitive and easy to navigate, making the application accessible to users of all technical skill levels.

By providing these features, the Personal Finance Manager aims to empower users to take control of their finances, make informed financial decisions, and ultimately achieve their financial goals. Whether you are looking to monitor your daily spending, plan for future expenses, or simply gain a better understanding of your financial situation, the Personal Finance Manager is the perfect tool to help you on your journey to financial well-being.

## Objectives

The primary objectives of the Personal Finance Manager project are:

1. **Expense Tracking**:
   * Enable users to log and categorize their daily expenses.
   * Provide a user-friendly interface for viewing and managing expenses over different time periods (daily, weekly, monthly).
2. **Budget Management**:
   * Allow users to set and manage budgets for various expense categories.
   * Track actual spending against the set budgets and notify users when they are nearing or exceeding their budget limits.
3. **Financial Reporting**:
   * Generate detailed financial reports to give users insights into their spending patterns.
   * Use visual tools such as charts and graphs to help users better understand their financial data and make informed decisions.
4. **User Authentication and Data Security**:
   * Implement secure user registration and login functionalities to protect user data.
   * Ensure that all user information, including passwords, is securely stored and managed.
5. **User-Friendly Interface**:
   * Develop an intuitive graphical user interface (GUI) using Tkinter to make the application accessible and easy to navigate for users of all technical skill levels.
   * Provide clear and straightforward interactions for users to manage their finances effortlessly.
6. **Data Integrity and Reliability**:
   * Utilize MySQL for reliable and secure data storage.
   * Ensure data integrity by implementing robust database management practices and error handling mechanisms.
7. **Scalability and Future Enhancements**:
   * Design the application architecture to be scalable, allowing for future enhancements and additional features.
   * Maintain a modular code structure to facilitate easy updates and maintenance.

By achieving these objectives, the Personal Finance Manager aims to provide a comprehensive and effective tool for users to manage their personal finances, promoting better financial habits and helping users achieve their financial goals.

## Reasons for Developing the Project

The development of the Personal Finance Manager project is driven by several important reasons:

1. **Financial Awareness**:
   * Many individuals struggle with keeping track of their daily expenses and often lose sight of their financial situation. This project aims to provide users with a clear understanding of their spending habits, promoting financial awareness and responsible financial behaviour.
2. **Budget Management**:
   * Budgeting is a crucial aspect of personal finance that helps individuals allocate their income towards essential expenses, savings, and discretionary spending. This project assists users in setting and maintaining budgets, ensuring they can effectively manage their finances and avoid overspending.
3. **Expense Tracking**:
   * Keeping track of expenses can be tedious and time-consuming. By automating the expense tracking process, the Personal Finance Manager simplifies this task, making it easier for users to record and categorize their expenses accurately.
4. **Data-Driven Decisions**:
   * Access to detailed financial reports and visualizations enables users to make informed financial decisions. By analyzing their spending patterns and budget performance, users can identify areas where they can cut costs and save more effectively.
5. **User-Friendly Interface**:
   * Many existing financial management tools are complex and difficult to use. This project focuses on creating an intuitive and user-friendly interface, making it accessible to users of all technical skill levels and ensuring a positive user experience.
6. **Secure Data Management**:
   * Security is a significant concern when it comes to managing personal finances. The project emphasizes secure user authentication and data protection, ensuring that users' financial information is stored safely and securely.
7. **Customization and Flexibility**:
   * Different users have different financial needs and goals. The Personal Finance Manager allows for customization in tracking various categories of expenses and budgets, providing flexibility to cater to individual user requirements.
8. **Promoting Financial Health**:
   * Ultimately, the project aims to promote better financial health by helping users gain control over their finances. By providing tools for effective financial management, the project encourages users to adopt healthier financial practices and work towards achieving their financial goals.
9. **Educational Value**:
   * The project also serves an educational purpose by teaching users about financial management principles. Through hands-on experience with budgeting and expense tracking, users can develop essential financial skills that benefit them in the long run.

By addressing these reasons, the Personal Finance Manager project aspires to be a valuable tool that empowers users to take charge of their personal finances, leading to improved financial well-being and stability.

## Goals

The Personal Finance Manager project aims to achieve the following goals:

1. **Enhanced Financial Tracking**:
   * Develop a reliable system for users to log and categorize their daily expenses accurately.
   * Ensure that users can easily access and review their financial transactions over various time periods.
2. **Effective Budget Management**:
   * Provide users with the tools to set and manage budgets for different expense categories.
   * Implement features that help users monitor their spending against their budgets, promoting better financial discipline.
3. **Comprehensive Financial Reporting**:
   * Create detailed and insightful financial reports that offer users a clear understanding of their spending habits.
   * Utilize visual tools such as charts and graphs to make the financial data more accessible and easier to interpret.
4. **Secure User Authentication**:
   * Implement a robust user authentication system to ensure that user data is protected and access is restricted to authorized users only.
   * Encrypt sensitive information, such as passwords, to enhance security.
5. **Intuitive User Interface**:
   * Design a user-friendly graphical user interface (GUI) using Tkinter that is easy to navigate for users of all technical skill levels.
   * Ensure that the application interface is clean, intuitive, and responsive.
6. **Data Integrity and Reliability**:
   * Utilize MySQL for reliable and secure data storage.
   * Ensure the integrity of the financial data through robust database management practices and thorough error handling.
7. **Customization and Flexibility**:
   * Allow users to customize their expense categories and budgets to fit their unique financial needs and preferences.
   * Provide flexibility in generating and viewing financial reports based on user-defined criteria.
8. **User Education and Empowerment**:
   * Educate users about effective financial management practices through hands-on interaction with the application.
   * Empower users to take control of their personal finances and make informed financial decisions.
9. **Scalability and Future Enhancements**:
   * Design the application architecture to be scalable, allowing for easy integration of additional features and future enhancements.
   * Maintain a modular code structure to facilitate updates and improvements without disrupting the overall functionality.
10. **Promotion of Financial Health**:
    * Encourage users to adopt healthier financial habits by providing them with the tools and insights necessary for effective financial management.
    * Aim to improve users' overall financial well-being and stability through consistent use of the application.

By striving to meet these goals, the Personal Finance Manager project seeks to provide a comprehensive, user-friendly, and secure solution for managing personal finances, ultimately helping users achieve better financial health and control.

## Advantages

The Personal Finance Manager project offers several key advantages that make it a valuable tool for users seeking to manage their personal finances effectively:

1. **Simplified Expense Tracking**:
   * Users can easily log and categorize their expenses, which helps in keeping track of where their money is going. This simplification reduces the effort required to monitor daily spending.
2. **Effective Budget Management**:
   * The application allows users to set and manage budgets for various expense categories. This feature helps users control their spending, avoid overspending, and save money more effectively.
3. **Insightful Financial Reporting**:
   * The application generates detailed financial reports, providing users with a clear understanding of their spending habits. Visual tools like charts and graphs make it easier to analyse financial data and identify trends.
4. **Enhanced Financial Awareness**:
   * By regularly logging expenses and reviewing reports, users become more aware of their financial habits. This increased awareness can lead to more responsible financial behaviour and better financial decisions.
5. **User-Friendly Interface**:
   * The graphical user interface (GUI) designed with Tkinter is intuitive and easy to navigate. Users of all technical skill levels can quickly learn to use the application, making financial management accessible to everyone.
6. **Secure Data Management**:
   * The application includes secure user authentication and data protection mechanisms. This ensures that users' financial information is stored safely and that their privacy is maintained.
7. **Customization and Flexibility**:
   * Users can customize expense categories and budgets according to their specific needs and preferences. This flexibility ensures that the application can cater to a wide range of personal finance scenarios.
8. **Data Integrity and Reliability**:
   * Utilizing MySQL for data storage ensures that the financial data is stored reliably and can be accessed efficiently. Robust database management practices help maintain data integrity.
9. **Time-Saving**:
   * Automating the expense tracking and budgeting processes saves users’ time that would otherwise be spent manually recording and analysing their finances.
10. **Scalability**:
    * The application is designed to be scalable, allowing for future enhancements and additional features without major restructuring. This ensures that the application can grow with the users' needs.
11. **Educational Value**:
    * Using the application helps users learn about personal finance management principles. By interacting with the budgeting and expense tracking features, users can develop essential financial skills.
12. **Improved Financial Health**:
    * By providing tools for effective financial management, the application helps users improve their overall financial health. Users can achieve better control over their finances, reduce unnecessary spending, and increase savings.
13. **Accessibility**:
    * The application can be used by anyone with a basic understanding of computers and finance. Its design ensures that even users with limited technical skills can benefit from its features.

By offering these advantages, the Personal Finance Manager project aims to be an indispensable tool for individuals looking to manage their personal finances more efficiently and achieve their financial goals.

## Disadvantages

While the Personal Finance Manager project offers numerous benefits, there are also some potential disadvantages that users should be aware of:

1. **Learning Curve**:
   * Although the application is designed to be user-friendly, some users, especially those not familiar with technology or personal finance concepts, might find it challenging to learn how to use all features effectively.
2. **Data Entry Effort**:
   * Users need to manually enter their expenses and budgets into the system. This can be time-consuming and may lead to inaccuracies if users do not consistently and accurately log their financial transactions.
3. **Security Risks**:
   * Despite the implementation of security measures, storing sensitive financial information in a digital format always carries some risk. Users must trust that the application will safeguard their data against potential breaches.
4. **Dependence on Technology**:
   * Users need access to a computer and possibly the internet to use the application effectively. This can be a limitation for individuals who do not have regular access to these resources.
5. **Initial Setup Time**:
   * Setting up the application and inputting initial financial data, such as past expenses and current budget plans, can be time-consuming. This initial effort might deter some users from fully utilizing the application.
6. **Limited Financial Guidance**:
   * While the application helps track and manage finances, it does not provide personalized financial advice or guidance. Users still need to have a basic understanding of financial principles to make informed decisions.
7. **Potential for Data Loss**:
   * In the event of a system failure, software bug, or user error, there is a risk of losing financial data. Regular backups are essential to mitigate this risk, but users must remember to perform them.
8. **Customization Limitations**:
   * Although the application allows for some customization, it may not be flexible enough to meet all users' unique financial management needs. Users with complex financial situations might find the application lacking in advanced features.
9. **Updates and Maintenance**:
   * The application may require periodic updates and maintenance to fix bugs, improve security, and add new features. Users need to ensure they keep their software up-to-date, which can be an additional burden.
10. **Database Management Complexity**:
    * Managing a MySQL database may be complex for users who are not familiar with database management. Issues such as database corruption or connectivity problems can pose challenges.
11. **Platform Dependence**:
    * The application might be dependent on specific platforms or operating systems, limiting its usability for users with different systems. Cross-platform compatibility might be an issue.

By considering these disadvantages, users can make a more informed decision about whether the Personal Finance Manager application meets their needs and expectations. It also provides valuable insights for future development and improvement of the application to address these limitations.

## Hardware and Software Requirements

### Hardware Requirements

1. **Computer**:
   * A personal computer or laptop with a minimum of:
     + Processor: 1 GHz or faster
     + RAM: 2 GB or more
     + Hard Disk: At least 500 MB of free space
2. **Internet Connection**:
   * An internet connection is required for downloading necessary software and libraries, and for database connectivity if using a remote database server.
3. **Input Devices**:
   * Keyboard and mouse or other pointing devices for user interaction.

### Software Requirements

1. **Operating System**:
   * The application is compatible with major operating systems including:
     + Windows 7 or later
     + macOS 10.12 or later
     + Linux distributions with a compatible Python environment
2. **Python**:
   * Python 3.7 or later must be installed. The project is developed using Python, and it requires a compatible Python runtime environment.
3. **Python Libraries**:
   * Several Python libraries are required to run the application. These can be installed using the requirements.txt file:

mysql-connector-python

tk

hashlib

* + You can install all the required libraries using the following command:

pip install -r requirements.txt

1. **MySQL Server**:
   * MySQL Server should be installed and running either locally or on a remote server. This is used for storing and managing the application's data.
   * MySQL Workbench (optional) can be used for database management and query execution.
2. **Database Configuration**:
   * A MySQL database should be set up with the necessary tables and schemas. Ensure the database credentials are configured correctly in the application's connection settings.
3. **Integrated Development Environment (IDE)**:
   * An IDE or code editor such as PyCharm, Visual Studio Code, or any other preferred editor for writing and debugging the Python code.
4. **GUI Toolkit**:
   * Tkinter, which is included with standard Python installations, is used for developing the graphical user interface (GUI).
5. **Version Control System (optional)**:
   * Git for version control, allowing you to track changes and collaborate with others. GitHub or GitLab can be used for remote repository management.

### Example

mysql-connector-python

### Installing Required Libraries

To install the required libraries, run the following command in your project's root directory:

pip install -r requirements.txt

## Python: An Overview

### Python Programming LanguageIntroduction

Python is a high-level, interpreted programming language known for its simplicity and readability. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability and simplicity, making it an ideal choice for both beginners and experienced developers.

### Features of Python

1. **Readability**:
   * Python's syntax is clean and easy to understand, allowing developers to write code that is more readable and maintainable.
2. **Interpreted Language**:
   * Python code is executed line-by-line, which makes debugging easier and allows for interactive programming.
3. **Dynamically Typed**:
   * Variables in Python do not require an explicit declaration to reserve memory space. The interpreter automatically identifies the type of data.
4. **High-Level Language**:
   * Python abstracts the complexities of the computer's hardware, making it easier to program without worrying about the underlying details.
5. **Extensive Standard Library**:
   * Python comes with a rich standard library that supports many common programming tasks such as string manipulation, file handling, and internet protocols.
6. **Cross-Platform**:
   * Python runs on various platforms, including Windows, macOS, Linux, and Unix, making it a versatile choice for developers.
7. **Community Support**:
   * Python has a large and active community, which contributes to a vast ecosystem of libraries, frameworks, and tools.

### History of Python

* **Late 1980s**: Guido van Rossum begins working on Python as a hobby project.
* **February 1991**: The first version, Python 0.9.0, is released.
* **1994**: Python 1.0 is released with features like lambda, map, filter, and reduce.
* **2000**: Python 2.0 introduces list comprehensions and garbage collection.
* **2008**: Python 3.0 is released, focusing on removing redundant features and fixing design flaws in the language.
* **Present**: Python continues to evolve, with regular updates and an active development community.

### The Best Way to Learn Python – Python Programming Tutorial for BeginnersAdvantages of Python

1. **Easy to Learn and Use**:
   * Python's straightforward syntax and readability make it an excellent choice for beginners and for rapid development.
2. **Versatile**:
   * Python can be used for web development, data analysis, artificial intelligence, scientific computing, automation, and more.
3. **Large Standard Library**:
   * The extensive standard library provides modules and functions for various tasks, reducing the need for writing code from scratch.
4. **Integration Capabilities**:
   * Python can easily integrate with other languages and technologies, such as C, C++, Java, and .NET.
5. **Community and Support**:
   * A large, active community means plenty of resources, tutorials, and third-party modules are available.
6. **Productivity and Speed**:
   * Python's simplicity and the availability of numerous libraries enable developers to be more productive and write fewer lines of code compared to other languages.

### Applications of Python

1. **Web Development**:
   * Frameworks like Django, Flask, and Pyramid enable rapid development of robust web applications.
2. **Data Science and Machine Learning**:
   * Libraries such as NumPy, Pandas, Matplotlib, and Scikit-Learn make Python a popular choice for data analysis and machine learning.
3. **Automation and Scripting**:
   * Python is often used for writing scripts to automate repetitive tasks, making it a favorite tool for system administrators.
4. **Scientific Computing**:
   * Packages like SciPy and SymPy support scientific and mathematical computations.
5. **Artificial Intelligence and Deep Learning**:
   * Libraries such as TensorFlow, Keras, and PyTorch facilitate the development of AI and deep learning models.
6. **Software Development**:
   * Python can be used to develop software applications, from simple scripts to complex enterprise applications.
7. **Game Development**:
   * Libraries like Pygame enable the development of simple games and multimedia applications.

## Database: An Overview

### Most Popular Databases: Best Database to Work in 2020Introduction

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. Databases are managed by Database Management Systems (DBMS), which handle data storage, retrieval, and management. Databases enable users to efficiently store, modify, and retrieve data, supporting various applications ranging from simple data logging to complex enterprise systems.

### DBMS | Types of Databases - javatpointTypes of Databases

1. **Relational Databases (RDBMS)**:
   * Use structured tables with rows and columns.
   * Tables are related to each other through foreign keys.
   * Examples: MySQL, PostgreSQL, Oracle, Microsoft SQL Server.
2. **NoSQL Databases**:
   * Designed for unstructured or semi-structured data.
   * Types include document stores, key-value stores, wide-column stores, and graph databases.
   * Examples: MongoDB (document store), Radis (key-value store), Cassandra (wide-column store), Neo4j (graph database).
3. **In-Memory Databases**:
   * Store data in the system’s main memory (RAM) for fast access.
   * Examples: Radis, Me cached.
4. **NewSQL Databases**:
   * Combine the ACID guarantees of traditional RDBMS with the scalability of NoSQL systems.
   * Examples: Google Spanner, CockroachDB.

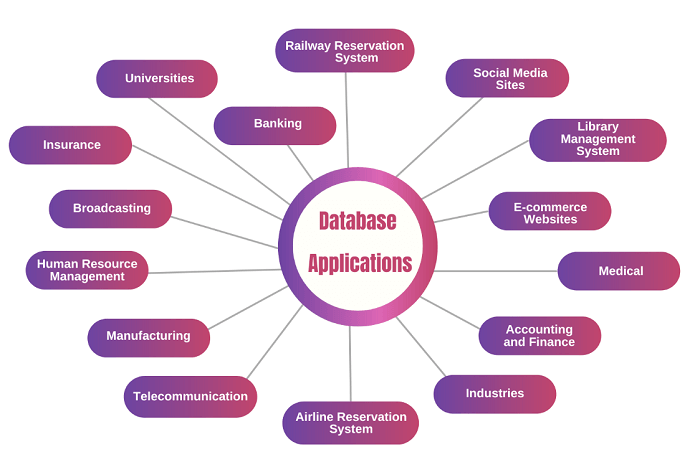
### Database Management System (DBMS): An Overview and Patentability Analysis | Blog Posts | LumenciFeatures of Databases

1. **Data Storage and Retrieval**:
   * Efficiently store and retrieve large amounts of data.
2. **Data Integrity**:
   * Ensure data accuracy and consistency through constraints and transactions.
3. **Data Security**:
   * Provide authentication and authorization mechanisms to secure data.
4. **Backup and Recovery**:
   * Support data backup and restoration to prevent data loss.
5. **Concurrency Control**:
   * Manage simultaneous data access by multiple users.
6. **Scalability**:
   * Support data and workload growth through horizontal or vertical scaling.

### Advantages of Using Databases

1. **Improved Data Management**:
   * Centralized data management allows for efficient handling and organization of data.
2. **Data Sharing**:
   * Multiple users can access and share data concurrently.
3. **Data Security**:
   * Protect sensitive data through robust security mechanisms.
4. **Data Integrity**:
   * Maintain data accuracy and consistency through integrity constraints and transactions.
5. **Data Backup and Recovery**:
   * Safeguard against data loss through regular backups and recovery options.
6. **Efficient Data Access**:
   * Optimize data retrieval through indexing and query optimization.

### Applications of Databases

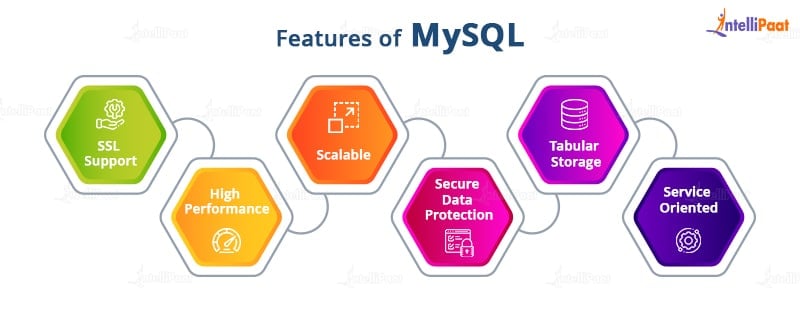
1. **Business Applications**:
   * Customer Relationship Management (CRM), Enterprise Resource Planning (ERP), and other business management systems.
2. **E-commerce**:
   * Online shopping platforms, inventory management, and order processing.
3. **Healthcare**:
   * Patient records management, medical research databases.
4. **Banking and Finance**:
   * Transaction processing, account management, and financial analysis.
5. **Education**:
   * Student information systems, online learning platforms.
6. **Social Media**:
   * User data management, content storage, and activity tracking.

## MySQL: An Overview

### MySQL (@MySQL) / XIntroduction

MySQL is an open-source relational database management system (RDBMS) based on Structured Query Language (SQL). It is widely used for web applications and is known for its reliability, ease of use, and performance. MySQL is developed, distributed, and supported by Oracle Corporation.

### Features of MySQL

1. **Open Source**:
   * MySQL is free to use and modify under the GNU General Public License.
2. **High Performance**:
   * Designed for speed and efficiency, capable of handling large databases.
3. **Cross-Platform**:
   * Runs on various operating systems including Windows, macOS, Linux, and Unix.
4. **Security**:
   * Provides robust security features such as user authentication, data encryption, and access control.
5. **Scalability**:
   * Can scale vertically (adding resources to a single server) and horizontally (distributing the database across multiple servers).
6. **Replication**:
   * Supports master-slave replication, enabling data replication across multiple servers for redundancy and load balancing.
7. **ACID Compliance**:
   * Ensures data integrity through Atomicity, Consistency, Isolation, and Durability (ACID) properties in transactions.
8. **Storage Engines**:
   * Supports multiple storage engines like InnoDB (default), MyISAM, and more, allowing customization based on specific use cases.

### OpenOcean - How MariaDB became the world-leading open-source database companyHistory of MySQL

* **1995**: MySQL was created by Swedish developers David Axmark, Allan Larsson, and Michael Widenius.
* **2000**: MySQL becomes popular as a part of the LAMP stack (Linux, Apache, MySQL, PHP/Perl/Python).
* **2008**: Sun Microsystems acquires MySQL AB, the company behind MySQL.
* **2010**: Oracle Corporation acquires Sun Microsystems, including MySQL.
* **Present**: MySQL continues to be actively developed and maintained by Oracle, with a strong community and enterprise support.

### Write features of My SQL. Answer... MP Board Class-12 Informatics Practices question answer collectionAdvantages of MySQL

1. **Ease of Use**:
   * Simple installation and user-friendly interfaces.
2. **Cost-Effective**:
   * Free and open-source, with paid enterprise options for additional features and support.
3. **Reliable and Stable**:
   * Proven track record of reliability and stability in various applications.
4. **Extensive Documentation**:
   * Comprehensive documentation and a large community provide ample resources for troubleshooting and learning.
5. **Integration**:
   * Easily integrates with various programming languages and development environments.

### Applications of MySQL

1. **Web Applications**:
   * Commonly used in web development for managing user data, content management, and e-commerce platforms.
2. **Content Management Systems (CMS)**:
   * Backbone for popular CMS platforms like WordPress, Joomla, and Drupal.
3. **E-commerce Platforms**:
   * Used by online retailers for managing products, orders, and customer data.
4. **Data Warehousing**:
   * Suitable for data warehousing applications due to its scalability and performance.
5. **Enterprise Applications**:
   * Supports business applications such as ERP and CRM systems.

## Python Modules | DigitalOceanModules Used in the Personal Finance Manager Project

### 1. mysql.connector

**Description**: This module is used to connect to a MySQL database server, execute SQL queries, and manage database transactions.

**Purpose in the Project**:

* Establishes a connection with the MySQL database.
* Executes SQL queries to perform CRUD (Create, Read, Update, Delete) operations on the database.

**Example Usage**:

import mysql.connector

connection = mysql.connector.connect(

host='localhost',

database='personal\_finance\_manager',

user='your\_username',

password='your\_password'

)

### 2. hashlib

**Description**: This module provides a common interface to many secure hash and message digest algorithms, including SHA-256.

**Purpose in the Project**:

* Used for hashing user passwords before storing them in the database to enhance security.

**Example Usage**:

import hashlib

def hash\_password(password):

return hashlib.sha256(password.encode()).hexdigest()

### 3. tkinter

**Description**: Tkinter is the standard GUI (Graphical User Interface) library for Python, which provides tools to create desktop applications.

**Purpose in the Project**:

* Used to create the graphical user interface for the Personal Finance Manager application.
* Handles user input, button clicks, and displays data.

**Example Usage**:

import tkinter as tk

root = tk.Tk()

root.title("Personal Finance Manager")

### 4. getpass

**Description**: This module provides a secure way to handle password input, preventing passwords from being displayed on the screen.

**Purpose in the Project**:

* Used during user registration and login processes to securely input passwords.

**Example Usage**:

from getpass import getpass

password = getpass("Enter your password: ")

### 5. os (Optional)

**Description**: The os module provides a way to interact with the operating system, allowing for file and directory management, environment variables, and more.

**Purpose in the Project** (if used):

* Manage configuration settings, environment variables, or file paths.
* Optional for enhanced functionality like reading database credentials from environment variables.

**Example Usage**:

import os

db\_user = os.getenv('DB\_USER')

db\_password = os.getenv('DB\_PASSWORD')

### Summary of Modules

* **mysql.connector**: Connects to and interacts with the MySQL database.
* **hashlib**: Hashes passwords for secure storage.
* **tkinter**: Creates the graphical user interface.
* **getpass**: Securely handles password input.
* **os**: Manages environment settings (optional).

## Programming languages: Python's new developer in residence and their 'make-it-or-break-it' role | ZDNETPersonal Finance Manager Project

### Database Configuration (db.py)

import mysql.connector

def connect():

return mysql.connector.connect(

host="localhost",

user="root",

password="admin@123",

database="your\_database"

)

def create\_tables():

connection = connect()

cursor = connection.cursor()

cursor.execute("""

CREATE TABLE IF NOT EXISTS Users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL

);

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS Expenses (

expense\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

amount DECIMAL(10, 2),

category VARCHAR(255),

date DATE,

description TEXT,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id)

);

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS Budgets (

budget\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

category VARCHAR(255),

amount DECIMAL(10, 2),

start\_date DATE,

end\_date DATE,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id)

);

""")

connection.commit()

cursor.close()

connection.close()

if \_\_name\_\_ == "\_\_main\_\_":

create\_tables()

### User Authentication (user.py)

from db import connect

import hashlib

import mysql.connector

def hash\_password(password):

return hashlib.sha256(password.encode()).hexdigest()

def register(username, password):

connection = connect()

cursor = connection.cursor()

try:

cursor.execute("""

INSERT INTO Users (username, password)

VALUES (%s, %s);

""", (username, hash\_password(password)))

connection.commit()

print("User registered successfully.")

except mysql.connector.Error as err:

print(f"Error: {err}")

finally:

cursor.close()

connection.close()

def login(username, password):

connection = connect()

cursor = connection.cursor()

try:

cursor.execute("""

SELECT user\_id, password FROM Users WHERE username = %s;

""", (username,))

user = cursor.fetchone()

if user and user[1] == hash\_password(password):

print("Login successful.")

return user[0]

else:

print("Invalid username or password.")

return None

except mysql.connector.Error as err:

print(f"Error: {err}")

return None

finally:

cursor.close()

connection.close()

### Expense Management (expense.py)

from db import connect

def add\_expense(user\_id, amount, category, date, description):

connection = connect()

cursor = connection.cursor()

cursor.execute("""

INSERT INTO Expenses (user\_id, amount, category, date, description)

VALUES (%s, %s, %s, %s, %s);

""", (user\_id, amount, category, date, description))

connection.commit()

cursor.close()

connection.close()

def view\_expenses(user\_id):

connection = connect()

cursor = connection.cursor()

cursor.execute("""

SELECT \* FROM Expenses WHERE user\_id = %s;

""", (user\_id,))

expenses = cursor.fetchall()

cursor.close()

connection.close()

return expenses

### Budget Management (budget.py)

from db import connect

def set\_budget(user\_id, category, amount, start\_date, end\_date):

connection = connect()

cursor = connection.cursor()

cursor.execute("""

INSERT INTO Budgets (user\_id, category, amount, start\_date, end\_date)

VALUES (%s, %s, %s, %s, %s);

""", (user\_id, category, amount, start\_date, end\_date))

connection.commit()

cursor.close()

connection.close()

def view\_budgets(user\_id):

connection = connect()

cursor = connection.cursor()

cursor.execute("""

SELECT \* FROM Budgets WHERE user\_id = %s;

""", (user\_id,))

budgets = cursor.fetchall()

cursor.close()

connection.close()

return budgets

### Report Generation (report.py)

from db import connect

def generate\_report(user\_id):

connection = connect()

cursor = connection.cursor()

cursor.execute("""

SELECT category, SUM(amount) as total

FROM Expenses

WHERE user\_id = %s

GROUP BY category;

""", (user\_id,))

report = cursor.fetchall()

cursor.close()

connection.close()

return report

### Graphical User Interface (gui.py)

import tkinter as tk

from tkinter import messagebox

from db import create\_tables

import user

import expense

import budget

import report

class FinanceManagerApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Personal Finance Manager")

self.user\_id = None

self.login\_screen()

def clear\_screen(self):

for widget in self.root.winfo\_children():

widget.destroy()

def login\_screen(self):

self.clear\_screen()

tk.Label(self.root, text="Username:").grid(row=0, column=0)

self.username\_entry = tk.Entry(self.root)

self.username\_entry.grid(row=0, column=1)

tk.Label(self.root, text="Password:").grid(row=1, column=0)

self.password\_entry = tk.Entry(self.root, show="\*")

self.password\_entry.grid(row=1, column=1)

tk.Button(self.root, text="Login", command=self.login).grid(row=2, column=0)

tk.Button(self.root, text="Register", command=self.register\_screen).grid(row=2, column=1)

def register\_screen(self):

self.clear\_screen()

tk.Label(self.root, text="Username:").grid(row=0, column=0)

self.reg\_username\_entry = tk.Entry(self.root)

self.reg\_username\_entry.grid(row=0, column=1)

tk.Label(self.root, text="Password:").grid(row=1, column=0)

self.reg\_password\_entry = tk.Entry(self.root, show="\*")

self.reg\_password\_entry.grid(row=1, column=1)

tk.Button(self.root, text="Register", command=self.register).grid(row=2, column=0)

tk.Button(self.root, text="Back", command=self.login\_screen).grid(row=2, column=1)

def login(self):

username = self.username\_entry.get()

password = self.password\_entry.get()

user\_id = user.login(username, password)

if user\_id:

self.user\_id = user\_id

self.user\_menu()

else:

messagebox.showerror("Error", "Invalid username or password")

def register(self):

username = self.reg\_username\_entry.get()

password = self.reg\_password\_entry.get()

user.register(username, password)

messagebox.showinfo("Success", "User registered successfully")

self.login\_screen()

def user\_menu(self):

self.clear\_screen()

tk.Button(self.root, text="Add Expense", command=self.add\_expense\_screen).grid(row=0, column=0)

tk.Button(self.root, text="View Expenses", command=self.view\_expenses).grid(row=1, column=0)

tk.Button(self.root, text="Set Budget", command=self.set\_budget\_screen).grid(row=2, column=0)

tk.Button(self.root, text="View Budgets", command=self.view\_budgets).grid(row=3, column=0)

tk.Button(self.root, text="Generate Report", command=self.generate\_report).grid(row=4, column=0)

tk.Button(self.root, text="Logout", command=self.login\_screen).grid(row=5, column=0)

def add\_expense\_screen(self):

self.clear\_screen()

tk.Label(self.root, text="Amount:").grid(row=0, column=0)

self.amount\_entry = tk.Entry(self.root)

self.amount\_entry.grid(row=0, column=1)

tk.Label(self.root, text="Category:").grid(row=1, column=0)

self.category\_entry = tk.Entry(self.root)

self.category\_entry.grid(row=1, column=1)

tk.Label(self.root, text="Date (YYYY-MM-DD):").grid(row=2, column=0)

self.date\_entry = tk.Entry(self.root)

self.date\_entry.grid(row=2, column=1)

tk.Label(self.root, text="Description:").grid(row=3, column=0)

self.description\_entry = tk.Entry(self.root)

self.description\_entry.grid(row=3, column=1)

tk.Button(self.root, text="Add Expense", command=self.add\_expense).grid(row=4, column=0)

tk.Button(self.root, text="Back", command=self.user\_menu).grid(row=4, column=1)

def add\_expense(self):

amount = float(self.amount\_entry.get())

category = self.category\_entry.get()

date = self.date\_entry.get()

description = self.description\_entry.get()

expense.add\_expense(self.user\_id, amount, category, date, description)

messagebox.showinfo("Success", "Expense added successfully")

self.user\_menu()

def view\_expenses(self):

self.clear\_screen()

expenses = expense.view\_expenses(self.user\_id)

for idx, exp in enumerate(expenses):

tk.Label(self.root, text=str(exp)).grid(row=idx, column=0)

tk.Button(self.root, text="Back", command=self.user\_menu).grid(row=len(expenses), column=0)

def set\_budget\_screen(self):

self.clear\_screen()

tk.Label(self.root, text="Category:").grid(row=0, column=0)

self.budget\_category\_entry = tk.Entry(self.root)

self.budget\_category\_entry.grid(row=0, column=1)

tk.Label(self.root, text="Amount:").grid(row=1, column=0)

self.budget\_amount\_entry = tk.Entry(self.root)

self.budget\_amount\_entry.grid(row=1, column=1)

tk.Label(self.root, text="Start Date (YYYY-MM-DD):").grid(row=2, column=0)

self.start\_date\_entry = tk.Entry(self.root)

self.start\_date\_entry.grid(row=2, column=1)

tk.Label(self.root, text="End Date (YYYY-MM-DD):").grid(row=3, column=0)

self.end\_date\_entry = tk.Entry(self.root)

self.end\_date\_entry.grid(row=3, column=1)

tk.Button(self.root, text="Set Budget", command=self.set\_budget).grid(row=4, column=0)

tk.Button(self.root, text="Back", command=self.user\_menu).grid(row=4, column=1)

def set\_budget(self):

category = self.budget\_category\_entry.get()

amount = float(self.budget\_amount\_entry.get())

start\_date = self.start\_date\_entry.get()

end\_date = self.end\_date\_entry.get()

budget.set\_budget(self.user\_id, category, amount, start\_date, end\_date)

messagebox.showinfo("Success", "Budget set successfully")

self.user\_menu()

def view\_budgets(self):

self.clear\_screen()

budgets = budget.view\_budgets(self.user\_id)

for idx, bud in enumerate(budgets):

tk.Label(self.root, text=str(bud)).grid(row=idx, column=0)

tk.Button(self.root, text="Back", command=self.user\_menu).grid(row=len(budgets), column=0)

def generate\_report(self):

self.clear\_screen()

report\_data = report.generate\_report(self.user\_id)

for idx, row in enumerate(report\_data):

tk.Label(self.root, text=str(row)).grid(row=idx, column=0)

tk.Button(self.root, text="Back", command=self.user\_menu).grid(row=len(report\_data), column=0)

if \_\_name\_\_ == "\_\_main\_\_":

create\_tables()

root = tk.Tk()

app = FinanceManagerApp(root)

root.mainloop()

### Main Entry Point (main.py)

import tkinter as tk

from gui import FinanceManagerApp

from db import create\_tables

if \_\_name\_\_ == "\_\_main\_\_":

create\_tables()

root = tk.Tk()

app = FinanceManagerApp(root)

root.mainloop()

### Future Scope of the Personal Finance Manager Project

1. **Integration with Banking APIs**: Enhance the application by integrating it with banking APIs to automatically import transactions. This will help users keep their expenses up-to-date without manual input.
2. **Mobile Application Development**: Develop mobile versions of the application for Android and iOS to allow users to manage their finances on the go.
3. **Advanced Analytics and AI**: Incorporate machine learning algorithms to provide personalized financial advice, detect spending patterns, and predict future expenses.
4. **Multi-Currency Support**: Expand the application to support multiple currencies, making it suitable for users who travel frequently or have international expenses.
5. **Investment Tracking**: Add features to track investments, including stocks, bonds, mutual funds, and cryptocurrencies, providing users with a comprehensive view of their financial portfolio.
6. **Bill Reminders and Management**: Implement a bill reminder system to notify users of upcoming payments and due dates, reducing the chances of missed payments and late fees.
7. **Collaborative Budgeting**: Allow multiple users to collaborate on a shared budget, useful for families or roommates managing joint expenses.
8. **Enhanced Security Features**: Strengthen security by adding features such as two-factor authentication, biometric login, and end-to-end encryption for sensitive financial data.
9. **Data Visualization**: Improve data visualization with more advanced charts and graphs, helping users to better understand their financial data and trends.
10. **Customizable Reports**: Allow users to generate customizable reports based on different time frames, categories, and financial metrics for more detailed financial analysis.
11. **Expense Categorization Automation**: Utilize machine learning to automatically categorize expenses based on transaction descriptions and user behaviour.
12. **Community and Support Forums**: Create a platform for users to share tips, ask questions, and get support from the community and financial experts.
13. **Third-Party Service Integration**: Integrate with third-party services such as tax preparation software, financial planning tools, and accounting services to provide a seamless financial management experience.
14. **Subscription Management**: Add features to manage and track subscriptions, helping users to avoid unwanted or forgotten subscriptions and manage recurring payments efficiently.
15. **Localization and Language Support**: Offer the application in multiple languages and localize it to suit the cultural and regulatory needs of users from different regions.

By implementing these future enhancements, the Personal Finance Manager project can evolve into a comprehensive financial management tool, catering to a wide range of user needs and preferences.

**Conclusion**

The Personal Finance Manager project represents a robust solution for individuals seeking to effectively manage their finances through a user-friendly interface. By enabling users to track expenses, set budgets, and generate insightful reports, this application promotes financial awareness and responsibility.

Throughout the development process, key functionalities such as user authentication, database management, and GUI implementation were meticulously integrated. These components ensure a seamless user experience, allowing for intuitive navigation and efficient management of financial data.

Looking ahead, the project offers ample opportunities for expansion and enhancement. Future iterations could incorporate advanced features like automated expense categorization, integration with banking APIs for real-time transaction updates, and predictive analytics using machine learning algorithms. Such enhancements would further empower users with personalized financial insights and proactive management capabilities.

In conclusion, the Personal Finance Manager project not only fulfills its core objective of facilitating financial management but also lays the groundwork for continued innovation and refinement. It stands poised to evolve alongside user needs and technological advancements, making a meaningful impact on personal financial well-being.

### References

1. **Python Programming Language**:
   * Python Software Foundation. (n.d.). Python.org. Retrieved from <https://www.python.org/>
2. **MySQL Database**:
   * Oracle Corporation. (n.d.). MySQL. Retrieved from <https://www.mysql.com/>
3. **Tkinter GUI Toolkit**:
   * Python Software Foundation. (n.d.). Tkinter Documentation. Retrieved from <https://docs.python.org/3/library/tkinter.html>
4. **Hashlib Library**:
   * Python Software Foundation. (n.d.). hashlib — Secure hashes and message digests. Retrieved from <https://docs.python.org/3/library/hashlib.html>
5. **MySQL Connector/Python**:
   * Oracle Corporation. (n.d.). MySQL Connector/Python Developer Guide. Retrieved from <https://dev.mysql.com/doc/connector-python/en/>
6. **Financial Management Concepts**:
   * Investopedia. (n.d.). Financial Management. Retrieved from https://www.investopedia.com/terms/f/financial-management.asp

These references provide foundational knowledge and documentation used during the development of the Personal Finance Manager project, ensuring accuracy and reliability in implementation.