# Personal Finance Application - Modular Version


A comprehensive personal finance management application built with Python and ttkbootstrap GUI framework.
This modular version breaks down the original monolithic code into organized, maintainable modules.

> **Recommended Python Version:** 3.10+ (tested on 3.11)

## Features

- **User Authentication**: Secure user registration and login with password hashing
- **Expense Management**: Add, view, edit, and delete expenses with categories
- **Income Tracking**: Manage income sources with categorization
- **Budget Overview**: Calculate budget summaries for custom date ranges
- **CSV Import/Export**: Import financial data from CSV files
- **Account Management**: Change passwords and view login history
- **Multi-user Support**: Each user has their own isolated financial data

## Project Structure

```plaintext
Honorable_Honey_Badger
├── __init__.py
├── main.py                     # Application entry point
├── README.md                   # Technical Documentation
├── requirements.txt            # Dependencies
├── data/                       # Databases
│   ├── finance.db              # Financial Data
│   └── users.db                # User Data
├── Docs/                       # Documentation
│   ├── [ISTM 601 - Group 8 Python (Group Contract) - Phase 1.pdf]
│   └── [ISTM 601 - Group 8 User Manual - Phase 2.pdf]
│   └── [ISTM 601 - Group 8 Technical Manual - Phase 2.pdf]
├── gui/                        # GUI pages
│   ├── __init__.py
│   ├── account_pages.py        # Account management
│   ├── auth.py                 # User Register/Logon
│   ├── budget_pages.py         # Budget Reporting
│   ├── expense_pages.py        # User Expense Page
│   ├── income_pages.py         # User Income Page
│   └── main_app.py             # Main Page
├── images/                     # Images, icons, CSS, etc.
│   └── HHB_1.png               # Login Page Image
└── operations/                 # Logic layer
    ├── __init__.py
    ├── database.py             # Database setup and operations
    └── finance_operations.py   # Finance DB Operations
```

## Installation

1. **Clone or download the project files**

2. **Install required dependencies:**

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

3. **Run the application:**

   * **From Command Line:**

     ```bash
     python main.py
     ```
   * **From Python IDLE:**

     1. Open IDLE (Python GUI)
     2. Go to **File → Open...** and select `main.py`
     3. Click **Run → Run Module** (or press **F5**) to start the application

## Module Descriptions

### `database.py`
Handles all database operations including:
- User database setup (authentication, login history)
- Finance database setup (expenses, income, categories)
- User registration and authentication functions
- Login history tracking

### `finance_operations.py`
Contains the `FinanceOperations` class that manages:
- Adding/loading expense and income categories
- Adding expenses and income entries
- Retrieving user financial data
- CSV import functionality
- Budget calculations

### `auth.py`
Contains the `AuthWindow` class for user authentication:
- Login/registration GUI
- Password validation
- Session management

### `expense_pages.py`
Contains the `ExpensePages` class managing:
- Expense entry forms
- Expense viewing and deletion
- CSV import interface
- Category management

### `income_pages.py`
Contains the `IncomePages` class managing:
- Income entry forms
- Income viewing and deletion
- CSV import interface
- Income category management

### `budget_pages.py`
Contains the `BudgetPages` class for:
- Budget overview calculations
- Date range selection
- Financial summaries

### `account_pages.py`
Contains the `AccountPages` class for:
- Password change functionality
- Login history display
- Account management

### `main_app.py`
Contains the `MainApp` class that:
- Creates the main application window
- Manages navigation between different sections
- Coordinates all page modules

### `main.py`
The application entry point that:
- Initializes databases
- Starts the authentication window
- Coordinates the application flow

## Usage

### First Time Setup
1. Run the application with `python main.py`
2. Click "Register" to create a new user account
3. Enter a username and password
4. Click "Login" to access the main application

### Managing Expenses
1. Click "Expenses" in the sidebar
2. Choose from manual entry, CSV import, viewing, or removing expenses
3. For manual entry, fill in all required fields and select/create categories
4. Use the date picker for accurate date selection

### Managing Income
1. Click "Income" in the sidebar
2. Similar workflow to expenses with income-specific categories
3. Track different income sources (salary, investments, etc.)

### Budget Overview
1. Click "Budget" in the sidebar
2. Select start and end dates for your budget period

3. Click "Calculate Budget" to see totals and savings

### CSV Import Format
Both expense and income CSV files should have these columns:
- `date` (YYYY-MM-DD format)
- `category` (text)
- `amount` (numeric)
- `description` (optional, text)

## Database Files
The application creates two SQLite database files:
- `users.db`: Stores user accounts and login history
- `finance.db`: Stores financial data (expenses, income, categories)

## Security Features
- Passwords are hashed using bcrypt with salt
- User data is isolated per account
- Session-based authentication

## Customization
- Add new expense/income categories through the GUI
- Modify themes by changing the `themename` parameter in window creation
- Extend functionality by adding new page modules

## Dependencies
- **pandas**: For CSV file processing
- **bcrypt**: For secure password hashing
- **ttkbootstrap**: For modern GUI components

## Troubleshooting

### Common Issues
1. **Import Error**: Make sure all dependencies are installed via `pip install -r requirements.txt`
2. **Database Errors**: Delete existing `.db` files to reset the database
3. **GUI Issues**: Ensure ttkbootstrap is properly installed

### Development Notes
- Each module is designed to be independent and reusable
- The original monolithic structure has been preserved in functionality
- Error handling has been centralized in the operations classes
- GUI components are separated from business logic

## Future Enhancements
- Add reporting and visualization features
- Implement data export functionality
- Add recurring transaction support
- Create backup/restore functionality
- Add transaction search and filtering