# BudgetWise AI-Based Expense Analyzer (Documentation)

## I. Project Overview: AI-Enabled Financial Analysis

The BudgetWise Expense Analyzer (Milestone 2) is a Streamlit web application designed to track and forecast personal expenses. It fulfills all requirements for Milestones 1 and 2 by integrating robust data management, secure user authentication, automatic data categorization, dynamic visualizations, and an AI-powered predictive model.

## II. Technical Architecture and File Structure

The project is built around a standard Python application structure using local CSV files for persistence.

| **File Name** | **Role in Project** | **Key Functionality** |
| --- | --- | --- |
| **app.py** | **Core Logic, UI, Forecasting** | Manages the Streamlit dashboard flow, navigation, filtering, charting, and executes the LinearRegression model. |
| **data\_utils.py** | **Data I/O & Persistence** | Handles safe reading/writing of data to local CSV files, ensuring data integrity and graceful failure handling. |
| **auth.py** | **Authentication Security** | Manages user registration and login using **bcrypt** for password hashing. |
| **categorizer.py** | **Expense Categorization (AI)** | Contains the logic for automatic categorization of transactions using keyword matching and regular expressions, defaulting to "Other" when no match is found. |
| **requirements.txt** | **Dependencies** | Lists all required Python packages for the project. |

## III. Project Timeline and Execution

This timeline represents the critical development phases for completing Milestones 1 and 2.

| Phase | Duration | Core Tasks Completed |
| --- | --- | --- |
| Phase 1: Foundation & Structure | Initial Setup (Estimated 3 Days) | Established project structure and dependencies (requirements.txt). Implemented basic data handling (data\_utils.py) and file integrity checks (ensure\_files()). |
| Phase 2: Security (Milestone 1) | Core Implementation (Estimated 5 Days) | Developed secure authentication (auth.py) using bcrypt hashing. Implemented Login Gatekeeping to secure the main dashboard. |
| Phase 3: Core Logic & Input | Feature Integration (Estimated 4 Days) | Integrated the Add Transaction form. Implemented the Auto-Categorization logic (categorizer.py) using keywords and import re. Fixed data persistence issues using pd.concat(). |
| Phase 4: Reporting & Filtering | Dashboard Development (Estimated 6 Days) | Integrated Summary Metrics and Visual Reports (Bar/Pie/Line Charts). Implemented Dynamic Filtering by Date, Category, and Type. Added Data Export functionality. |
| Phase 5: AI Forecasting (Milestone 2) | Advanced Implementation (Estimated 5 Days) | Integrated LinearRegression model for monthly expense forecasting. Implemented R² Score and Slope for model validation. Finalized robust date handling to prevent runtime errors. |

## IV. Milestone 1: Security and Data Foundation

Milestone 1 focused on creating the secure foundation for the application, ensuring data integrity and protecting user access.

### A. Secure User Authentication

• Bcrypt Hashing: User passwords are never stored in plaintext. The auth.py module uses the bcrypt library to generate a one-way hash of the password, which is stored in users.csv, ensuring the original password is never saved.  
• Access Control: The main dashboard features are protected by Login Gatekeeping, ensuring only authenticated users can access their transaction data and analysis tools.

### B. Robust Data Handling

• File Integrity (data\_utils.py): The ensure\_files() function guarantees that necessary data files exist and have the correct starting schema.  
• Safe Saving: The save\_transaction() and add\_user() functions use pd.concat() instead of the deprecated .append() method for modern, reliable, and secure appending of new data to the CSV files.

## V. Milestone 2: Features, Reporting, and AI

### A. Dynamic Data Handling and Categorization

• Auto-Categorization: The categorizer.py module uses a keyword map and regular expressions (import re) to scan the transaction description and automatically assign a category. If no match is found, the category is labeled 'Other'.  
• CSV Upload (Simple Mode): The application supports loading external transaction data, requiring the file strictly adheres to the schema for immediate analysis.

### B. Visual Analysis and Utility

• Summary Metrics: Visualization of Total Spent, Total Income, and Average Transaction Amount.  
• Dynamic Filtering: Users can apply filters for Date Range, Category, and Type to refine the data.  
• Visual Reports: The application dynamically generates Bar/Pie Charts and a Line Chart to visualize trends and distribution.  
• Data Export: An 'Export Filtered Data as CSV' button allows the user to download the current view.

### C. AI Expense Forecasting

• Model Implementation: Imports and uses LinearRegression (from scikit-learn) to model the time series of monthly spending totals.  
• Predictive Output: Calculates and displays the Forecasted Expense for the next calendar month.  
• Validation Metrics: The system includes metrics to validate the predictive model's performance:  
 - Trend Strength (R² Score): Measures the goodness-of-fit.  
 - Monthly Trend (Slope): Shows the calculated average monthly change in spending.