DAYANANDA SAGAR UNIVERSITY

SCHOOL OF ENGINEERING

Devarakaggalahalli, Harohalli, Kanakapura Road, Dt. Ramanagara, Karnataka 562112.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(DATA SCIENCE)

PROJECT REPORT

ON

"Money Management (Budget Buddy)"

2024-2025

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING (DATA SCIENCE)

Submitted By

Kammala Kalyan -ENG23DS0063

Kovouru Venkata Naga Sai Pranav -ENG23DS0067

Amarnath Gowda K M -ENG23DS0052

Under the Supervision of:

Ms. Prapti Bhattacharjee

Assistant Professor

COMPUTER SCIENCE &ENGINEERING (DATA SCIENCE)

DAYANANDA SAGAR
UNIVERSITY

School of Engineering

Department of Computer Science & Engineering

Harohalli, Ramanagara - 562112

Karnataka, India

CERTIFICATE

This is to certify that the DBMS MINI PROJECT titled "**Money Management**" carried out by **Sai Pranav(ENG23DS0067), Kammala Kalyan (ENG23DS0063), Amarnath Gowda K M(ENG23DS0052),** bona fide students of Bachelor of Technology in Computer Science and Engineering atthe School of Engineering, Dayananda Sagar University, Bangalore in partial fulfillment for the award of degree in Bachelor of Technology in Computer Science and Engineering, during the year 2023-2024.

Ms. Prapti Bhattacharjee
Asst. Professor
Dept. of CSE-DS
School of Engineering
Dayananda Sagar University

Dr. Shaila S G
Chairman, CSE-DS
School of Engineering
Dayananda Sagar
University

Dr. Uday Kumar Reddy
Dean
School of Engineering
Dayananda Sagar
University

DATE:

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this DBMS MINI PROJECT

First, we take this opportunity to express our sincere gratitude to the School of Engineering, Dayananda Sagar University for providing us with a great opportunity to pursue our bachelor's degree in this institution.

We would like to thank **Dr. Uday Kumar Reddy K R, Dean**, **School of Engineering**, **Dayananda Sagar University For** his constant encouragement and expert advice. It is immense pleasure to express our sincere thanks to **Dr. Shaila SG, Chairman**, **Department of Computer Science,and Engineering (Data Science)**, **Dayananda Sagar University,** for providing the right academicguidance that made our task possible.

We would like to thank our teacher **Ms. Prapti Bhattacharjee, Asst. Professor**, **Department of Computer Science and Engineering (Data Science)**, **Dayananda Sagar University**, for sparing her valuable time to extend help in every step of our DBMS MINI PROJECT, which paved the way forsmooth progress and the fruitful culmination of the project.

We are also grateful to our family and friends who provided us with every requirement throughout the course. We would like to thank one and all who directlyor indirectly helped us in the DBMS MINI PROJECT.

ABSTRACT

The Money Management project is a comprehensive database-driven solution designed to help users effectively manage their personal finances. Utilizing the principles of Database Management Systems (DBMS), this project enables secure storage, retrieval, and analysis of financial data including income, expenses, savings, and budget plans.

The backend is powered by a relational database ensuring data integrity, consistency, and scalability, while the frontend provides an intuitive user interface for seamless interaction.

By leveraging DBMS concepts such as normalization, SQL querying, indexing, and relational schema design, the project demonstrates how structured data management can simplify and enhance everyday money management tasks.

This project therefore helps to understand the concept of DBMS better and implement it in different applications of day-to-day life.

TABLE OF CONTENTS

PAGE NO

INTRODUCTION

Managing personal finances can be tough—especially when you're trying to stick to a budget but have no clear system to guide your spending. That's where our project, **Budget Buddy**, comes in. Designed with simplicity and usability in mind, Budget Buddy is a web-based application that helps users take control of their money with ease.

In this project, we were challenged to build a full-stack web application backed by a database. Our idea was inspired by a common problem many people face: tracking expenses, setting budgets, and knowing whether they're overspending—all in one place.

Budget Buddy allows users to:

- Set monthly budgets by category,
- Add and organize daily expenses,
- Monitor their spending in real time,
- And receive helpful insights through visual reports.

Built using **Flask** for the frontend and **MySQL** for the backend, our app delivers a smooth and intuitive experience. Whether you're saving up for something big or just trying to spend smarter, Budget Buddy gives users the clarity they need to make better financial decisions.

PROBLEM STATEMENT

Managing personal finances effectively can be challenging, as users often struggle with budgeting, expense tracking, and maintaining financial discipline. Many individuals overspend or fail to allocate their budgets appropriately due to a lack of real-time tracking and intuitive insights.

The goal of **Budget Buddy** is to provide users with a simple yet efficient web application that helps them:

- Set monthly budgets based on different spending categories.
- Track their expenses in real-time and compare against planned budgets.
- Maintain a financial streak by successfully staying within set limits.
- View detailed expense reports and insights through visual analytics.

By integrating a database-driven approach using **MySQL** for data storage and **Flask** for frontend interaction, Budget Buddy ensures seamless user experiences and helps individuals make informed financial decisions.

PROJECT DESCRIPTION

Budget Buddy is a user-friendly web application designed to help individuals manage their finances efficiently. The primary goal is to provide users with an intuitive platform to set budgets, track expenses, and analyze spending habits with real-time updates.

Key Features:

- **User Registration & Authentication**: Secure login for personalized financial management.
- **Budget Creation**: Users can set monthly budgets and categorize expenses.
- **Expense Tracking**: Allows users to log daily expenses and check spending progress.
- **Dashboard Overview:** Provides a real-time summary of budget usage.
- **Visual Reports**: Generates analytics and graphical reports to show spending trends.
- **Auto-Reset:** Budgets renew automatically based on user preferences.
- **Profile Management:** Users can update personal details and security settings.

Technology Stack:

- **Frontend**: Flask (lightweight and interactive UI)
- **Database**: MySQL (efficient storage and retrieval of financial data)

DESIGN

1. Frontend Design:

Technology: Flask (HTML, CSS, JavaScript)

- **Homepage**: Simple introduction with login/register options.
- **Dashboard**: Displays budget summary, expenses, and trends.
- **Forms**: User-friendly budget setup, expense logging, and category organization.
- **Charts**: Interactive graphs for financial analysis (e.g., bar charts for spending trends).
- **Responsive Design**: Optimized for both desktop and mobile users.

2. Backend Architecture:

Technology: Python (Flask), MySQL

- **Authentication Module**: Secure user login/registration with password hashing.
- **Budget Management**: CRUD operations for budgets, categories, and expenses.
- **Expense Tracking**: Real-time updates when users log expenses.
- **Reports**: SQL queries to generate summaries on spending history.

3. Database Schema:

Tables:

- **Users**: Stores login credentials and personal details.
- **Budgets**: Tracks budget cycles, planned vs. spent amounts.

- **Categories**: Organizes expenses into groups.
- **Expenses**: Logs all transactions with timestamps and category mapping.

4. User Flow:

- Login/Register → Authentication process.
- Dashboard → View budgets, expenses, and financial insights.
- Create Budget → Set categories and spending limits.
- Log Expenses → Enter transactions with category selection.
- Track Progress → Compare spending with budgeted amounts.
- Reports & History → Visual analysis and past trends.

5. Security Measures:

- **Data Encryption**: Protects sensitive user information.
- **SQL Injection Prevention**: Ensures safe database operations.

DATABASE DESIGN:

1. **Users**

Columns

i. id (INT, PK)
ii. username (VARCHAR)
iii. email (VARCHAR)
iv. password (VARCHAR)
v. streak (INT) — count of consecutive under-budget months

Notes

i. Each row is one registered user.
ii. id is referenced by all other tables to tie data back to the user.

2. Budgets

Columns

i. id (INT, PK)
ii. user_id (INT, FK → users.id)
iii. start_date, end_date (DATE) — budget period
iv. total_budget (DECIMAL) — the user's planned total

v. used_budget (DECIMAL) — running sum of actual spend
vi. is_active (TINYINT) — flags the current month's budget

Relationships

i. **1 user → N budgets** (a user can have many monthly budgets, but only one active at a time).

3. Categories

Columns

i. id (INT, PK)
ii. budget_id (INT, FK → budgets.id)
iii. name (VARCHAR) — e.g. "Food", "Transport"
iv. expected_amount (DECIMAL) — portion of total_budget
v. spent_amount (DECIMAL) — tally of expenses in this category

Relationships

- **1 budget → N categories** (each budget is broken into multiple categories).

4. Expenses

Columns

i. id (INT, PK)
ii. user_id (INT, FK → users.id)
iii. budget_id (INT, FK → budgets.id)
iv. category_id (INT, FK → categories.id)
v. expense_date (DATE)
vi. amount (DECIMAL)
vii. note (VARCHAR)

Relationships

i. **1 user → N expenses**
ii. **1 budget → N expenses**
iii. **1 category → N expenses**

Every expense row belongs to exactly one user, one budget period, and one category within that budget.

METHODOLOGY

The development of Budget Buddy follows a structured methodology to ensure an efficient and user-friendly financial management tool. The methodology consists of the following phases:

1. Requirement Analysis

- Identified user needs: Simple budget tracking, expense logging, and financial analytics.
- Defined system objectives: Secure authentication, dynamic budget management, and intuitive reporting.
- Conducted market research on similar applications to refine features.

2. System Design

- Architecture: Follows a client-server model with Flask as the backend and MySQL as the database.
- Database Schema: ER-diagram designed to structure users, budgets, categories, and expenses efficiently.
- UI/UX Design: Created wireframes ensuring intuitive navigation for financial tracking.
- Security Considerations: Implemented encryption for passwords and secure session handling.

3. Implementation & Development

- Frontend: Developed using HTML, CSS, and Flask templates for responsiveness.
- Backend: Used Python (Flask) to handle user requests, session management, and database operations.
- Database Management: Designed SQL queries for CRUD operations on budgets, expenses, and reports.
- Testing: Conducted unit testing for each module, ensuring data integrity and preventing SQL injection.

4. Testing & Validation

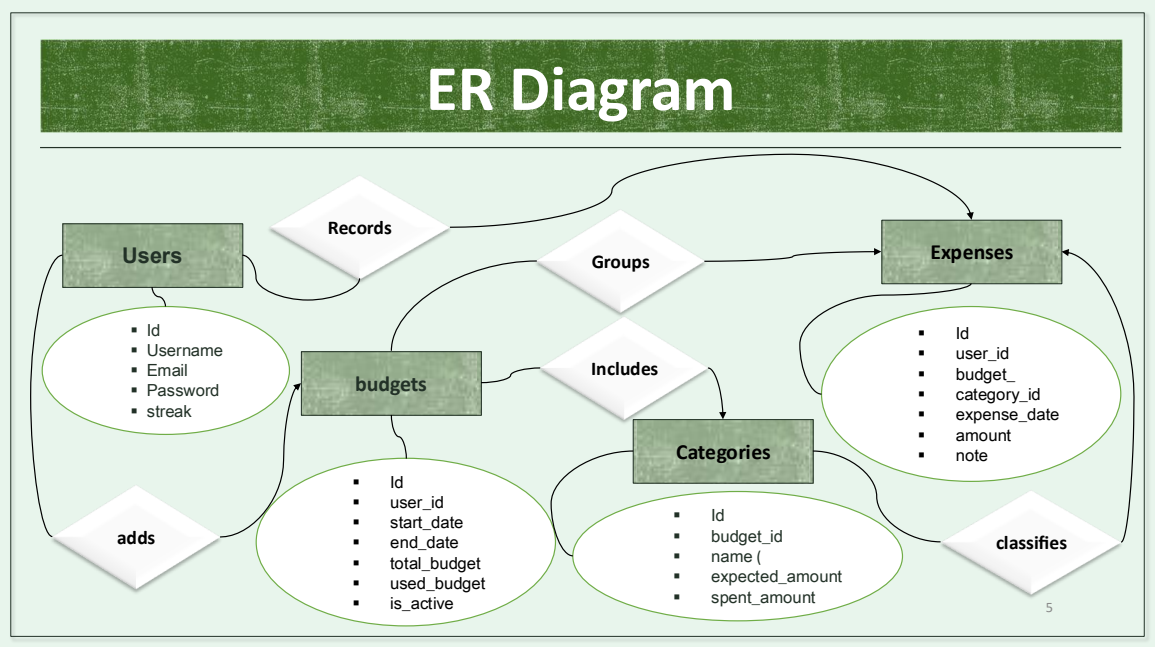
- Functional Testing: Verified that all features (budget creation, expense logging, and analytics) work as expected.
- Security Testing: Checked vulnerabilities related to authentication and data access.
- Performance Testing: Optimized queries to ensure fast response times.

5. Deployment & Maintenance

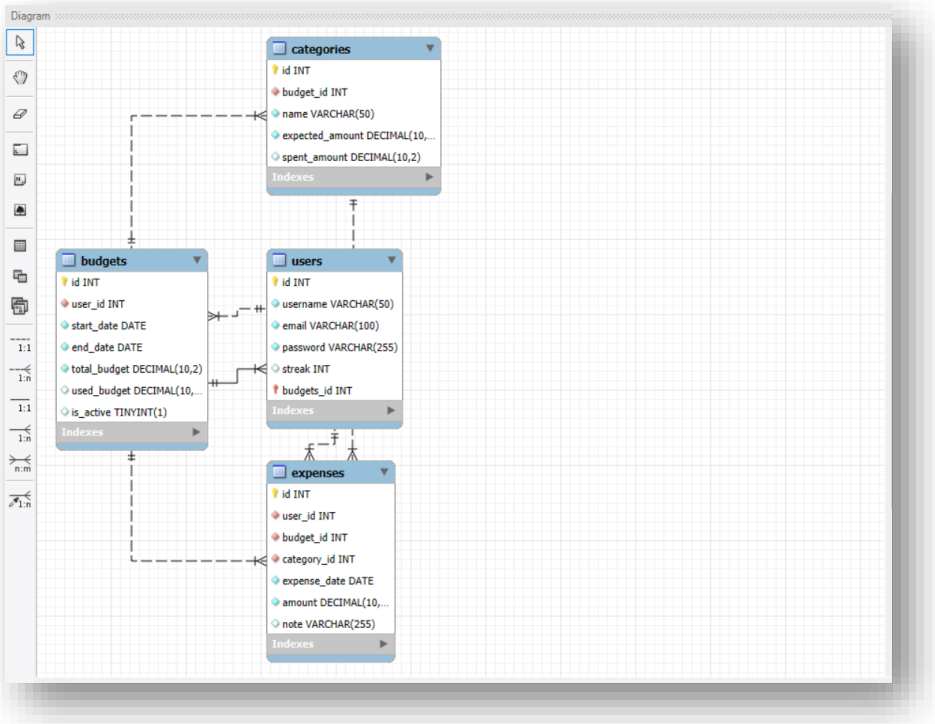
- Deployment: Hosted on a cloud-based platform for user accessibility.
- User Feedback: Gathered insights from beta testers to refine usability.
- Continuous Improvement: Regular updates based on usage patterns and user needs**.**

TESTING AND RESULT:

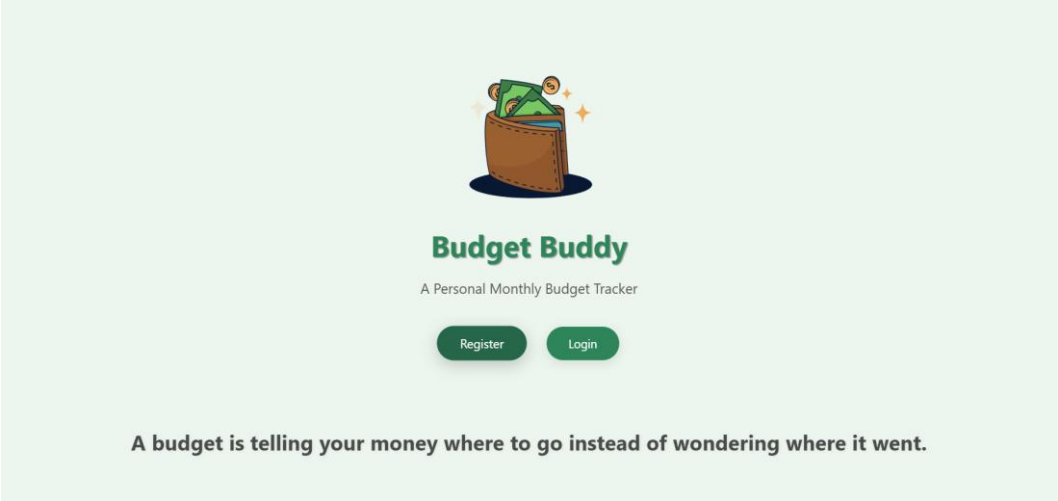
ER Diagram:

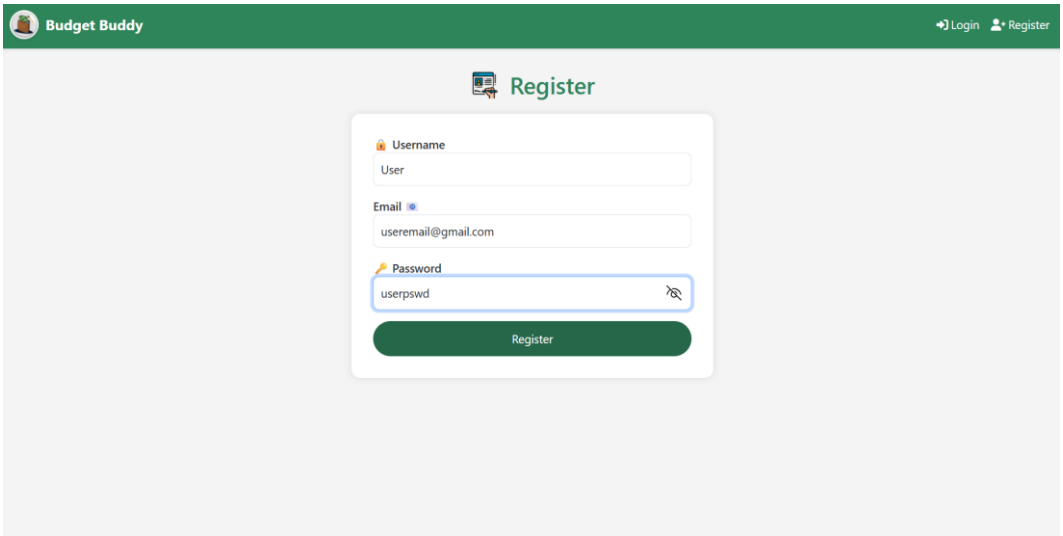


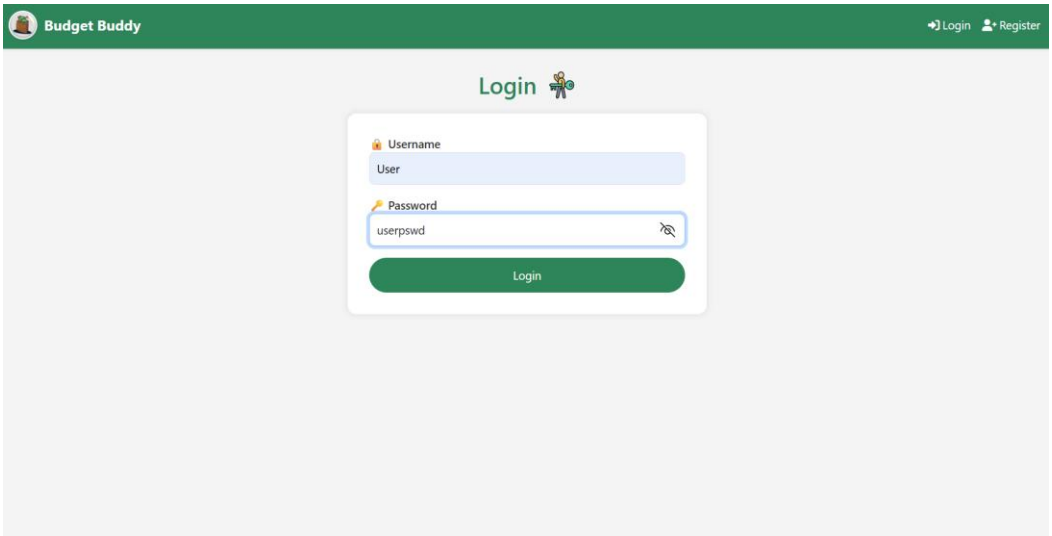
Schema Diagram :

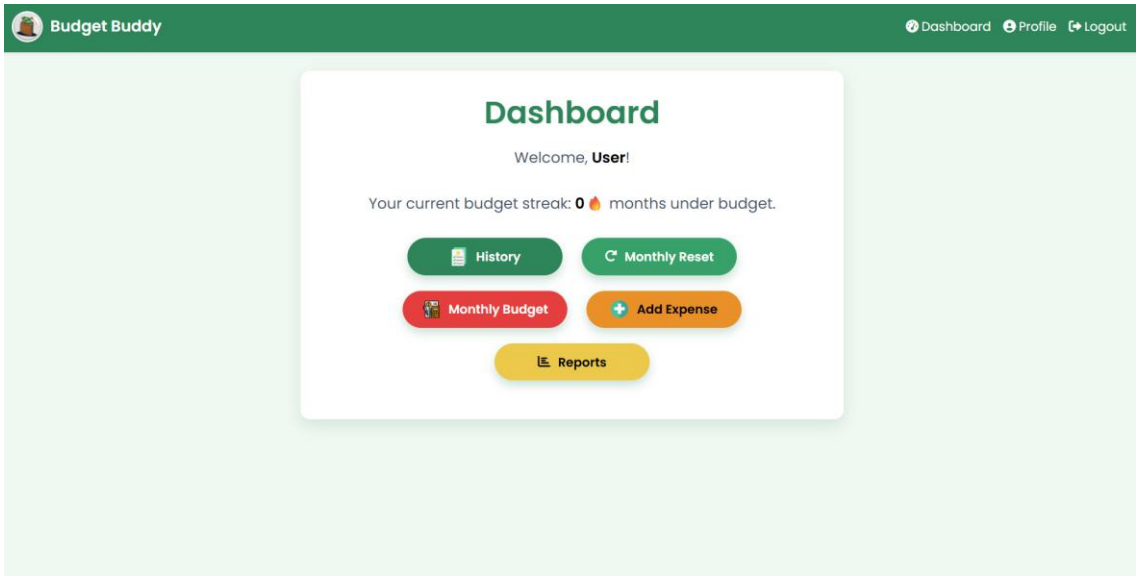


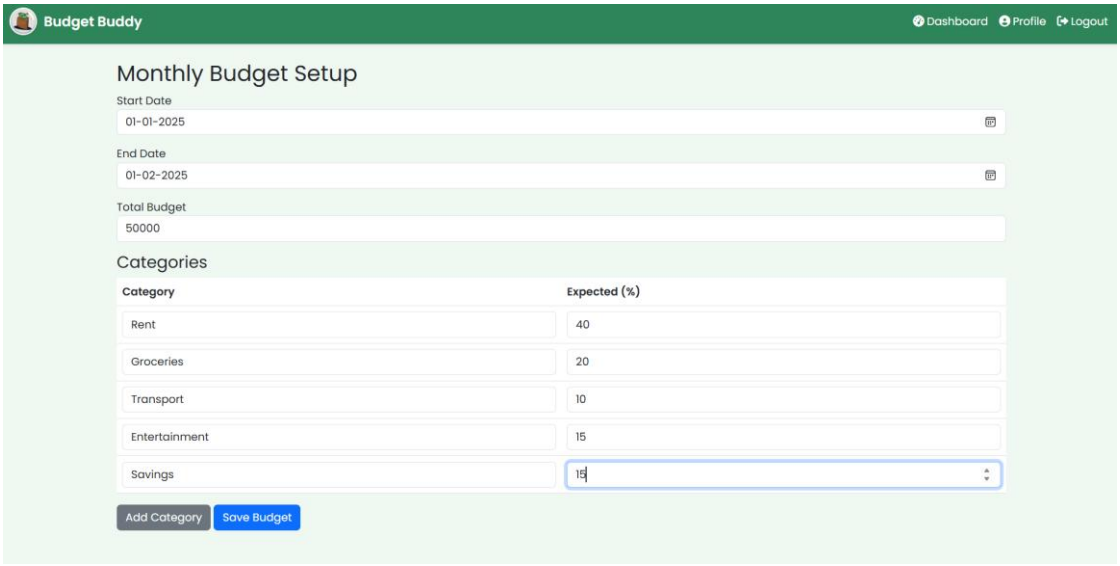
Result

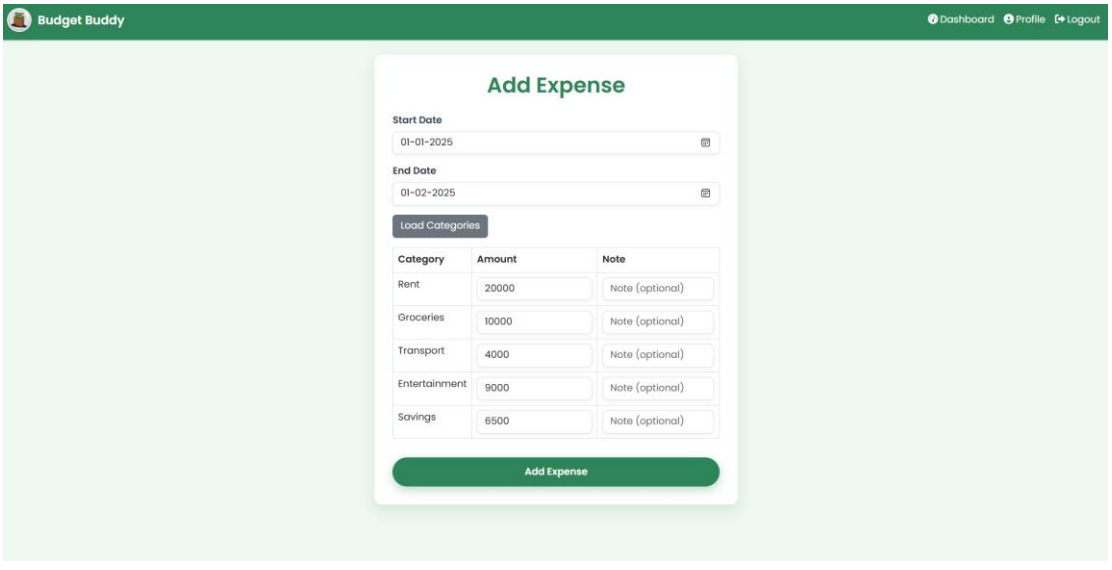

Budget Buddy
A Personal Monthly Budget Tracker
Register
Login
A budget is telling your money where to go instead of wondering where it went.


Budget Buddy
Login
Register
Register
Username
User
Email
useremail@gmail.com
Password
userpswd
Register

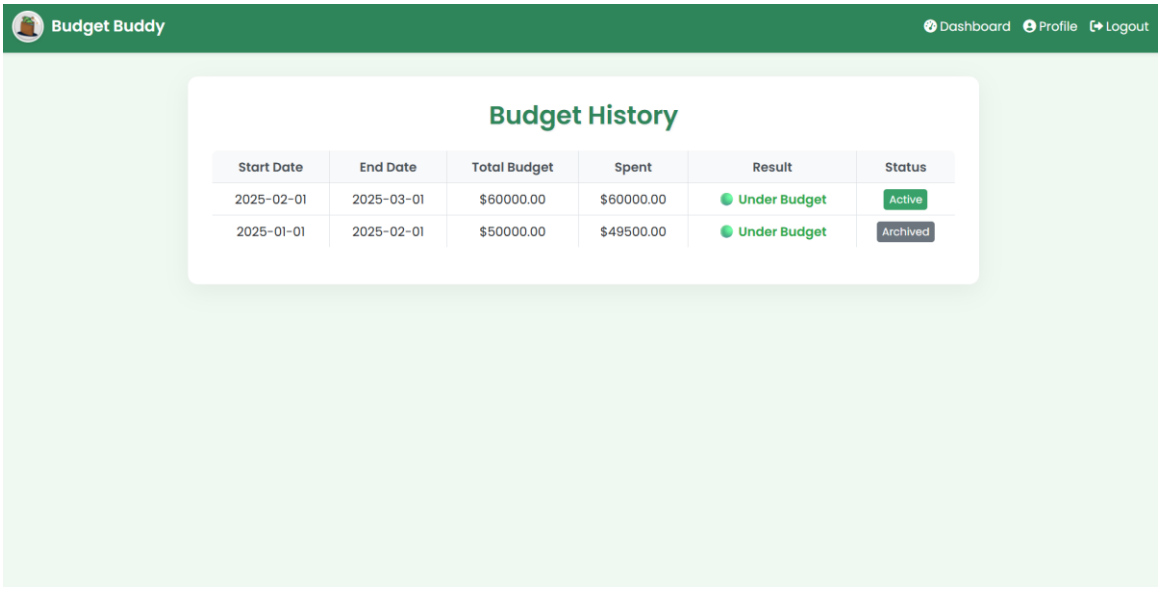

Budget Buddy
Login
Register
Login
Username
User
Password
userpswd
Login


Budget Buddy
Dashboard
Profile
Logout
Dashboard
Welcome, User!
Your current budget streak: 0 months under budget.
History
Monthly Reset
Monthly Budget
Add Expense
Reports

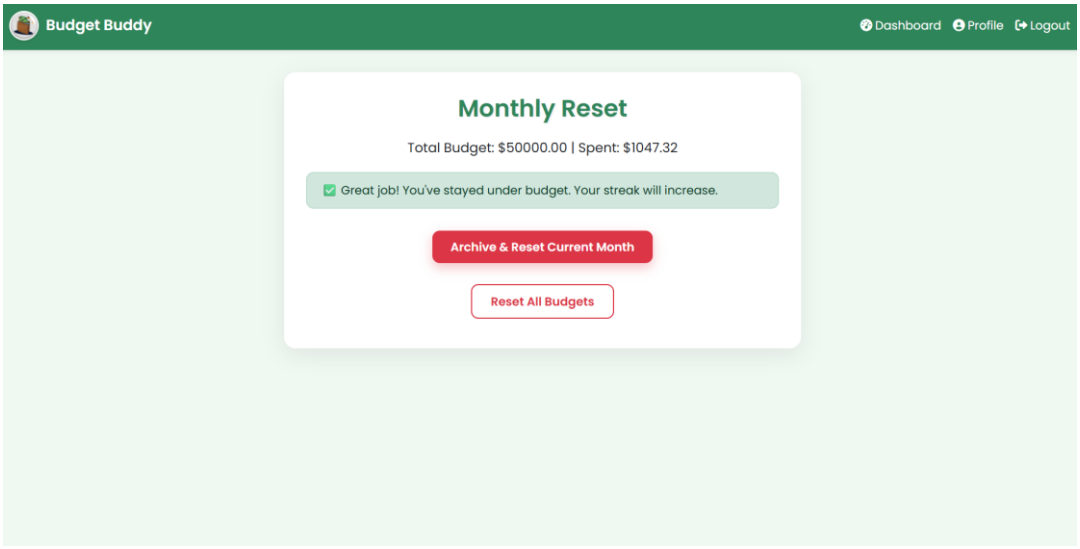

Budget Buddy
Dashboard
Profile
Logout
Monthly Budget Setup
Start Date
01-01-2025
End Date
01-02-2025
Total Budget
50000
Categories
Category
Expected (%)
Rent
40
Groceries
20
Transport
10
Entertainment
15
Savings
15
Add Category
Save Budget

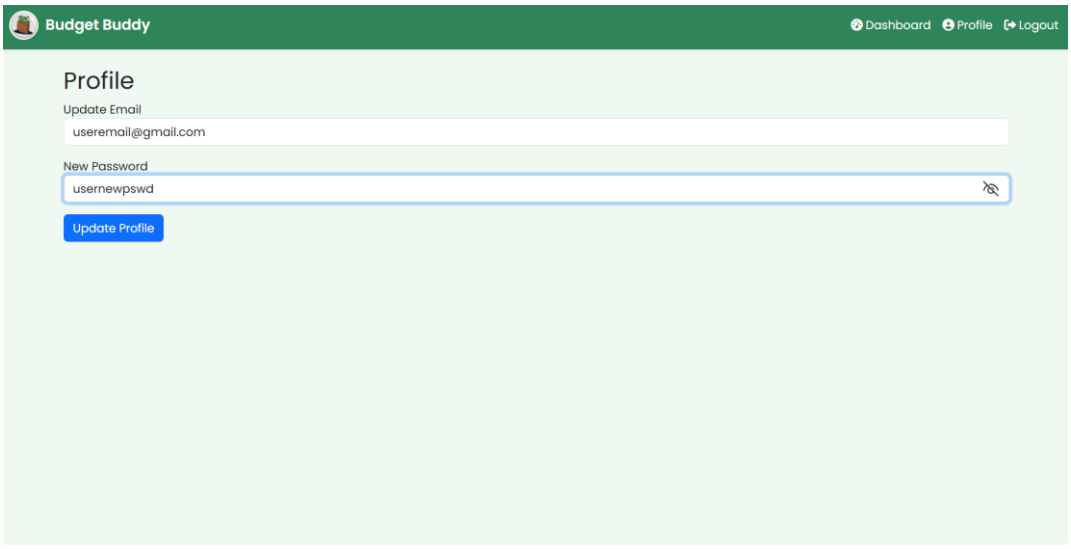

Budget Buddy
Dashboard
Profile
Logout
Add Expense
Start Date
01-01-2025
End Date
01-02-2025
Load Categories
Category
Amount
Note
Rent
20000
Note (optional)
Groceries
10000
Note (optional)
Transport
4000
Note (optional)
Entertainment
9000
Note (optional)
Savings
6500
Note (optional)
Add Expense


Budget Buddy
Dashboard
Profile
Logout
Budget Progress
Total Budget: $50000.00 | Spent: $49500.00
Rent: $20000.00 / $20000.00
100.0%
Groeries: $10000.00 / $10000.00
100.0%
Transport: $4000.00 / $5000.00
80.0%
Entertainment: $9000.00 / $7500.00
120.0%
Savings: $6500.00 / $7500.00
86.7%


Budget Buddy
Dashboard
Profile
Logout
Budget History
Start Date
End Date
Total Budget
Spent
Result
Status
2025-02-01
2025-03-01
$60000.00
$60000.00
Under Budget
Active
2025-01-01
2025-02-01
$50000.00
$49500.00
Under Budget
Archived


Budget Buddy
Dashboard
Profile
Logout
Budget Period: 2025-02-01 to 2025-03-01
Expected vs Actual Spending
Expected
Actual
Rent
Groceries
Transport
Entertainment
Savings
Budget Period: 2025-01-01 to 2025-02-01
Expected vs Actual Spending
Expected
Actual
Rent
Groceries
Transport
Entertainment
Savings


Budget Buddy
Dashboard
Profile
Logout
Monthly Reset
Total Budget: $50000.00 | Spent: $1047.32
Great job! You've stayed under budget. Your streak will increase.
Archive & Reset Current Month
Reset All Budgets


Budget Buddy
Dashboard
Profile
Logout
Profile
Update Email
useremail@gmail.com
New Password
usernewpswd
Update Profile

SYSTEM IMPLEMENTATION

```
CREATE DATABASE budgetbuddy;
USE budgetbuddy;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    streak INT DEFAULT 0
);

CREATE TABLE budgets (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    total_budget DECIMAL(10,2) NOT NULL,
    used_budget DECIMAL(10,2) DEFAULT 0,
    is_active BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE categories (
    id INT AUTO_INCREMENT PRIMARY KEY,
    budget_id INT NOT NULL,
    name VARCHAR(50) NOT NULL,
    expected_amount DECIMAL(10,2) NOT NULL,
    spent_amount DECIMAL(10,2) DEFAULT 0,
    FOREIGN KEY (budget_id) REFERENCES budgets(id)
);

CREATE TABLE expenses (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    budget_id INT NOT NULL,
    category_id INT NOT NULL,
    expense_date DATE NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    note VARCHAR(255),
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (budget_id) REFERENCES budgets(id),
    FOREIGN KEY (category_id) REFERENCES categories(id)
);
```

Code for implementation:

```python
@app.route('/register', methods=['GET', 'POST'])
def register():
    msg = ''
    if request.method == 'POST':
        username = request.form['username']
        email    = request.form['email']
        password = request.form['password']
        if not username or not email or not password:
            msg = 'Please fill out all fields.'
        else:
            cursor = mysql.connection.cursor()
            cursor.execute("SELECT * FROM users WHERE username = %s OR email = %s", (username, email))
            if cursor.fetchone():
                msg = 'Account already exists!'
            else:
                hashed = generate_password_hash(password)
                cursor.execute(
                    "INSERT INTO users (username, email, password, streak) VALUES (%s, %s, %s, 0)",
                    (username, email, hashed)
                )
                mysql.connection.commit()
                msg = 'You have successfully registered!'
    return render_template('register.html', msg=msg)
```

```python
@app.route('/login', methods=['GET', 'POST'])
def login():
    msg = ''
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        cursor = mysql.connection.cursor()
        cursor.execute("SELECT * FROM users WHERE username = %s", (username,))
        account = cursor.fetchone()
        if account and check_password_hash(account[3], password):
            session['user_id'] = account[0]
            session['username'] = account[1]
            return redirect(url_for('dashboard'))
        else:
            msg = 'Invalid username or password.'
    return render_template('login.html', msg=msg)
```

```python
@app.route('/dashboard')
@login_required
def dashboard():
    user_id = session['user_id']
    cursor = mysql.connection.cursor()
    cursor.execute("SELECT streak FROM users WHERE id = %s", (user_id,))
    streak = cursor.fetchone()[0] or 0
    return render_template('dashboard.html', streak=streak)
```

```
# 4) Update user streak from last active budget
cursor.execute("""
    SELECT total_budget, used_budget
    FROM budgets
    WHERE user_id = %s AND is_active = TRUE
""", (user_id,))
last = cursor.fetchone()
if last:
    old_total, old_used = last
    cursor.execute("SELECT streak FROM users WHERE id = %s", (user_id,))
    current_streak = cursor.fetchone()[0] or 0
    new_streak = current_streak + 1 if old_used <= old_total else 0
    cursor.execute("UPDATE users SET streak = %s WHERE id = %s", (new_streak, user_id))
    mysql.connection.commit()

# 5) Deactivate previous budgets
cursor.execute("UPDATE budgets SET is_active = FALSE WHERE user_id = %s", (user_id,))
mysql.connection.commit()

# 6) Create new budget
cursor.execute(
    "INSERT INTO budgets (user_id, start_date, end_date, total_budget, is_active, used_budget) "
    "VALUES (%s, %s, %s, %s, TRUE, 0)",
    (user_id, start_date, end_date, total_budget)
)
```


```
# 4) Update user streak from last active budget
cursor.execute("""
    SELECT total_budget, used_budget
    FROM budgets
    WHERE user_id = %s AND is_active = TRUE
""", (user_id,))
last = cursor.fetchone()
if last:
    old_total, old_used = last
    cursor.execute("SELECT streak FROM users WHERE id = %s", (user_id,))
    current_streak = cursor.fetchone()[0] or 0
    new_streak = current_streak + 1 if old_used <= old_total else 0
    cursor.execute("UPDATE users SET streak = %s WHERE id = %s", (new_streak, user_id))
    mysql.connection.commit()

# 5) Deactivate previous budgets
cursor.execute("UPDATE budgets SET is_active = FALSE WHERE user_id = %s", (user_id,))
mysql.connection.commit()

# 6) Create new budget
cursor.execute(
    "INSERT INTO budgets (user_id, start_date, end_date, total_budget, is_active, used_budget) "
    "VALUES (%s, %s, %s, %s, TRUE, 0)",
    (user_id, start_date, end_date, total_budget)
)
```


```
cursor.execute(
    "INSERT INTO expenses (user_id, budget_id, category_id, expense_date, amount, note) "
    "VALUES (%s, %s, %s, %s, %s, %s)",
    (uid, budget_id, cat_id, d_start, amount, note)
)
cursor.execute(
    "UPDATE categories SET spent_amount = spent_amount + %s WHERE id = %s",
    (amount, cat_id)
)
cursor.execute(
    "UPDATE budgets SET used_budget = used_budget + %s WHERE id = %s",
    (amount, budget_id)
)
```

CONCLUSION

In this project, we successfully designed and developed **Budget Buddy**, a full-stack web application aimed at simplifying personal finance management. By combining the power of a well-structured **MySQL database** with an intuitive **Flask-based frontend**, we created a tool that allows users to plan monthly budgets, log daily expenses, and track their financial habits with ease.

The project not only helped us apply core **DBMS concepts** such as relational modeling, SQL querying, and data integrity, but also enhanced our understanding of **full-stack development**, including frontend-backend integration, session handling, and user experience design.

Through features like real-time budget updates, visual spending reports, and a motivational streak tracker, Budget Buddy makes financial planning both accessible and engaging. This hands-on experience taught us how technical solutions can directly address real-world problems, and further strengthened our skills in building scalable, user-centric applications.

Budget Buddy stands as a meaningful step in our journey as developers and a tool with real potential to help users spend smarter and live better.

REFERENCES

- **Flask Documentation** – https://flask.palletsprojects.com/
  For building server-side logic and routing for the web application.
- **MySQL Documentation** – https://dev.mysql.com/doc/
  For designing and managing the relational database.
- **W3Schools SQL Tutorial** – https://www.w3schools.com/sql/
  For learning and referencing SQL commands and queries.
- **Node.js Documentation** – https://nodejs.org/en/docs/
  For understanding server-side scripting, asynchronous operations, and backend enhancements.
- **Express.js (if used)** – https://expressjs.com/
  For creating RESTful APIs and handling routes in Node.js (if applicable).
- **Jinja2 Templating (Flask Templates)** – https://jinja.palletsprojects.com/
  F
- or rendering dynamic web pages with Python and Flask.
- **Chart.js Documentation** – https://www.chartjs.org/docs/latest/
  For integrating interactive visual reports into the dashboard.
- **MDN Web Docs (HTML/CSS/JavaScript)** – https://developer.mozilla.org/
  For frontend styling, scripting, and UI responsiveness.
- **GeeksforGeeks / Stack Overflow**
  For debugging, code examples, and resolving implementation challenges.