Expanse Tracker

```
from datetime import datetime


class Expense:
    """Class to represent an expense item with the following attributes:
    """

    def __init__(self, expense_id, date, category, description, amount):
        """Initialize the expense item with the given attributes.

        Args:
            expense_id (int): Unique ID of the expense.
            date (str): Date of the expense in the format 'YYYY-MM-DD'.
            category (str): Category of the expense.
            description (str): Description of the expense.
            amount (float): Amount of the expense.
        """
        self.expense_id = expense_id
        self.date = date
        self.category = category
        self.description = description
        self.amount = amount

    def __str__(self):
        """Return a string representation of the expense item.

        Returns:
            str: String representation of the expense item.
        """
        return f'Expense ID: {self.expense_id}\n Date: {self.date}\nCategory: {self.category}\nDescription: {self.description}\nAmou


class ExpanseCalculator:
    """Class to represent an expense calculator with the following methods.
    """

    def __init__(self):
        """Initialize the expense storage as an empty list.
        """
        self.expense_storage = list()

    def add_expenses(self, expense):
        """Add an expense item to the expense storage.

        Args:
            expense (Expense): Expense item to be added.
        """
        self.expense_storage.append(expense)
        print("Expense added successfully.")

    def update_expense(self, expense_id, new_expense):
        """Update an existing expense item with a new expense item.

        Args:
            expense_id (int): Unique ID of the expense to be updated.
            new_expense (Expense): New expense item to replace the existing one.
        """
        for i, expense in enumerate(self.expense_storage):
            if expense.expense_id == expense_id:
                self.expense_storage[i] = new_expense
                print("Expense updated successfully.")
                return
        print("No item found with the given expense ID.")
```

```
        print("No item found with the given expense ID.")

    def delete_expense(self, expense_id):
        """Delete an expense item from the expense storage.

        Args:
            expense_id (int): Unique ID of the expense to be deleted.
        Returns:
            None
        """
        for i, expense in enumerate(self.expense_storage):
            if expense.expense_id == expense_id:
                self.expense_storage.pop(i)
                print("Expense deleted successfully!")
                return
        print("No item found with the given expense ID.")

    def display_expense(self):
        """Display all the expenses in the expense storage.
        """
        if self.expense_storage:
            print("Current Expenses:")
            for expense in self.expense_storage:
                print(expense)
        else:
            print("No expenses found.")

    def categorize_expenses(self):
        """Categorize the expenses based on their category.

        Returns:
            dict: A dictionary with category as key and total amount as value.
        """
        category_dictionary = dict()

        for expense in self.expense_storage:
            if expense.category in category_dictionary:
                category_dictionary[expense.category] += expense.amount
            else:
                category_dictionary[expense.category] = expense.amount

        return category_dictionary

    def summarize_expense(self):
        """Summarize the total expenses in the expense storage.

        Returns:
            float: Total amount of all the expenses.
        """
        return sum(expense.amount for expense in self.expense_storage)

    def generate_summary_report(self):
        """Generate a summary report of the expenses.
        """
        print("\nExpense Summary Report:")

        categorized_expenses = self.categorize_expenses()

        for category, total in categorized_expenses.items():
            print(f"Category: {category} | Total Expenses: ${total:.2f}")

        total_expense = self.summarize_expense()

        print(f"\nTotal Expense: ${total_expense:.2f}")
```

```
class Authentication:
    """Class to handle user authentication.
    """
    user_dict = {"user": "user", "user1": "user1"}

    def authenticate_user(self, username, password):
        """authenticate_user method to check if the user is authenticated.

        Args:
            username (str): Username of the user.
            password (str): Password of the user.

        Returns:
            bool: True if the user is authenticated, False
        """
        if username in self.user_dict:
            return self.user_dict[username] == password
        return False


def cli():
    """Command Line Interface for the Expense Tracker application
    """
    expense_calc = ExpanseCalculator()

    while True:
        choice = input(
            '\nSelect an option: \n1. Add item \n2. Update item \n3. Delete item \n4. Display expenses \n5. Generate summary report

        if choice == '1':
            print("\nEnter the details of the expense:")
            try:
                expense_id = int(input("Enter unique ID for expense: "))
                date = input("Enter the date (YYYY-MM-DD): ")
                datetime.strptime(date, '%Y-%m-%d')
                category = input("Enter the category: ")
                description = input("Enter the description: ")
                amount = float(input("Enter the amount: "))

                expense = Expense(expense_id, date, category,
                                  description, amount)
                expense_calc.add_expenses(expense)
            except ValueError as error:
                print(f"Invalid input: {error}")

        elif choice == '2':
            print("\nEnter the details of the updated expense:")
            try:
                expense_id = int(input("Enter unique ID: "))

                date = input("Enter the date (YYYY-MM-DD): ")
                datetime.strptime(date, '%Y-%m-%d')
                category = input("Enter the category: ")
                description = input("Enter the description: ")
                amount = float(input("Enter the amount: "))

                new_expense = Expense(
                    expense_id, date, category, description, amount)

                expense_calc.update_expense(expense_id, new_expense)
            except ValueError as error:
                print(f"Invalid input: {error}.")

        elif choice == '3':
            try:
```

```
                expense_id = int(input("Enter the expense ID to delete: "))
                expense_calc.delete_expense(expense_id)
            except ValueError as error:
                print(f"Invalid input: {error}")

        elif choice == '4':
            expense_calc.display_expense()

        elif choice == '5':
            expense_calc.generate_summary_report()

        elif choice == '6':
            print("\nExiting....")
            break

        else:
            print("\nInvalid choice. Try Again")


def main():
    """Main function to run the Expense Tracker application.
    """
    print("\nWelcome to Expense Tracker")

    print("\nUser Authentication")
    username = input("Enter username: ")
    password = input("Enter password: ")
    authenticator = Authentication()

    if authenticator.authenticate_user(username, password):
        print("Login Successful")
        cli()

    else:
        print("\nInvalid Credentials, Try again.")


if __name__ == "__main__":
    main()
```