TITLE: **M-Pesa Spending Predictor using Machine Learning**

Group Members

SCT221-0207/2023 -- Marylyne Nashipae

SCT221-0181/2023 -- Brian Kyalo Kimanzi

SCT221-0255/2023 -- Gideon Kipamet Kaiyian

SCT221-0522/2022 -- Kaguchia Kago

SCT221-0397/2022 -- Katra Noor Soyan

SCT221-0668/2022 – Stephen Githua

**Objective:**

To develop a web-based machine learning application that analyzes M-Pesa transaction statements and predicts future spending based on user input.

**Project Description:**

The M-Pesa Spending Predictor is a machine learning-powered web application built with Flask and XGBoost. It allows users to input details of their M-Pesa transactions and get predictions of their expected spending. The system is trained using historical transaction data and uses a combination of preprocessing, encoding, scaling, and regression modeling techniques.

**Tools & Technologies:**

1. Python (Data processing, Model training)
2. Flask (API backend)
3. HTML/JavaScript (Frontend)
4. XGBoost (Regression model)
5. scikit-learn (Preprocessing, Evaluation)
6. pandas/NumPy (Data manipulation)
7. pickle (Model serialization)
8. LabelEncoder / StandardScaler
9. CORS / flask\_restful

**Data Used:**

Cleaned M-Pesa transaction CSV with the following features:

1. Receipt
2. transaction\_Day
3. Year
4. Month
5. Date
6. Weekday
7. Hour
8. Minute
9. Seconds
10. Transaction\_type
11. Transaction\_party
12. Transaction\_amount
13. paid\_in\_or\_Withdraw
14. Balance

**Feature Engineering:**

1. Extracted relevant time features from the date and time columns.
2. Filtered and encoded categorical variables using LabelEncoder.
3. Special handling of infrequent Transaction\_party values.
4. Applied feature scaling using StandardScaler.

**Model Training & Evaluation:**

1. Used XGBRegressor with hyperparameter tuning via GridSearchCV.

Evaluated model using:

1. Root Mean Squared Error (RMSE) 40.19
2. Mean Absolute Error (MAE) 24.71
3. R2 Score 0.47

**Flask API:**

/prediction - Accepts POST requests with transaction details and returns predicted spending.

/data - Returns historical transaction data in JSON format.

**Frontend:**

1. HTML form with fields for Transaction Amount, Type, Party, and Paid/Withdrawn.
2. JavaScript to submit data via Fetch API and render results.

**Model Saving:**

1. Serialized trained model, scaler, and encoder into .pkl files for use in the API.

Challenges & Solutions:

1. Handled unseen categorical values by mapping them to constant value eg balance 1000.
2. Addressed scaling issues by fitting the scaler only on training data.
3. Improved model performance via feature selection and parameter tuning.