# Team Members:

Nyamach Duop

Regina Mayiee

The goal for Spave online banking is to help users manage their banking finances online by keeping track of their spending habits, managing their earnings, and showing where funds are being spent in a quick and efficient manner through a website.

* Customers will be allowed to create an account. With every new customer, the bank will confirm the new account and then open it. Customers will also be given an option of both savings and checking accounts or one of the two.
  + Strong password will be used, this means special characters, capital letters, longer than 10 characters, capital letters must be used.
  + User authentication and authorization will be needed
* Cash, debit cards, digital wallets, and bank transfers will be used for payment processing
  + Customers can take cash out through a cashier at the bank, or an ATM
* Customers will be able to perform deposits, withdraws, and transfers.
  + Notification will also be given for each transaction taken out of an account, and when a deposit is made. A notification will also be given when account balances are low.
* Transaction histories will be recorded, and users will be able to access and assess their deposits, transaction histories, and account balances.

# Tools and Technologies

**IDE:** Visual Studio

**Web Framework:** ASP.NET Core MVC for web application

**Database:** SQL Server or PostgreSQL

**ORM:** Entity Framework Core

**Authentication:** ASP.NET Core Identity

**Frontend:** Bootstrap for responsive design, possibly React or Angular for more dynamic UI

**APIs:** Twilio for SMS notifications, SendGrid for email notifications

# Schedule:

## Week 1 (06/10-0/16)

* **Project Planning:**
  + Define project scope and requirements.
  + Assign roles and responsibilities.
* **Development Environment Setup:**
  + Set up version control (GitHub/GitLab).
  + Configure development environments (Visual Studio, database server).

**Database Design and Initial Backend Development**

* **Database Design:**
  + Design the database schema (tables for users, accounts, transactions, etc.).
  + Create the database and set up initial tables.
* **Backend Development:**
  + Set up ASP.NET Core project.
  + Implement user authentication and authorization (registration, login).

**Basic Frontend Development**

* **Frontend Setup:**
  + Set up front-end framework (Angular, React, or Blazor).
  + Create the initial project structure and configure routing.
* **Initial UI Components:**
  + Design and implement login and registration pages.
  + Set up basic navigation and layout.

## Week 2 (06/17-06/23)

**Account and Transaction Management**

* **Backend:**
  + Implement APIs for account management (create, read, update, delete).
  + Implement transaction management APIs (record transaction, fetch transaction history).
* **Frontend:**
  + Design and implement account management UI.
  + Design and implement transaction history UI.

**Payment Processing and Security Features**

* **Backend:**
  + Integrate payment processing methods (cash, debit cards, digital wallets, bank transfers).
  + Implement security features (encryption, two-factor authentication).
* **Frontend:**
  + Design and implement payment processing UI.
  + Implement security features on the frontend (2FA setup, encryption).

**Testing and Debugging**

* **Backend:**
  + Write unit tests for backend functionality (authentication, transactions).
  + Perform integration testing.
* **Frontend:**
  + Conduct UI testing and fix any layout issues.
  + Ensure the flow between different components is seamless.

## Week 3 (06/24 - 07/01)

**Financial Management Tools and Alerts**

* **Backend:**
  + Implement APIs for spending analysis, budgeting tools, and savings goals.
  + Set up a notification system for alerts and reminders.
* **Frontend:**
  + Design and implement UI for financial management tools (graphs, charts, budgeting).
  + Implement alerts and notifications UI.

**Customer Support and Reporting**

* **Backend:**
  + Implement chat support integration and help center resources.
  + Develop APIs for generating downloadable statements and tax reports.
* **Frontend:**
  + Design and implement customer support interface.
  + Implement downloadable statements and tax report UI.

**Final Testing and Deployment Preparation**

* **Final Testing:**
  + Conduct thorough end-to-end testing.
  + Perform user acceptance testing (UAT) with sample users.
* **Deployment Preparation:**
  + Set up CI/CD pipeline.
  + Prepare deployment environment (Azure, AWS).
  + Deploy the application and perform final checks.