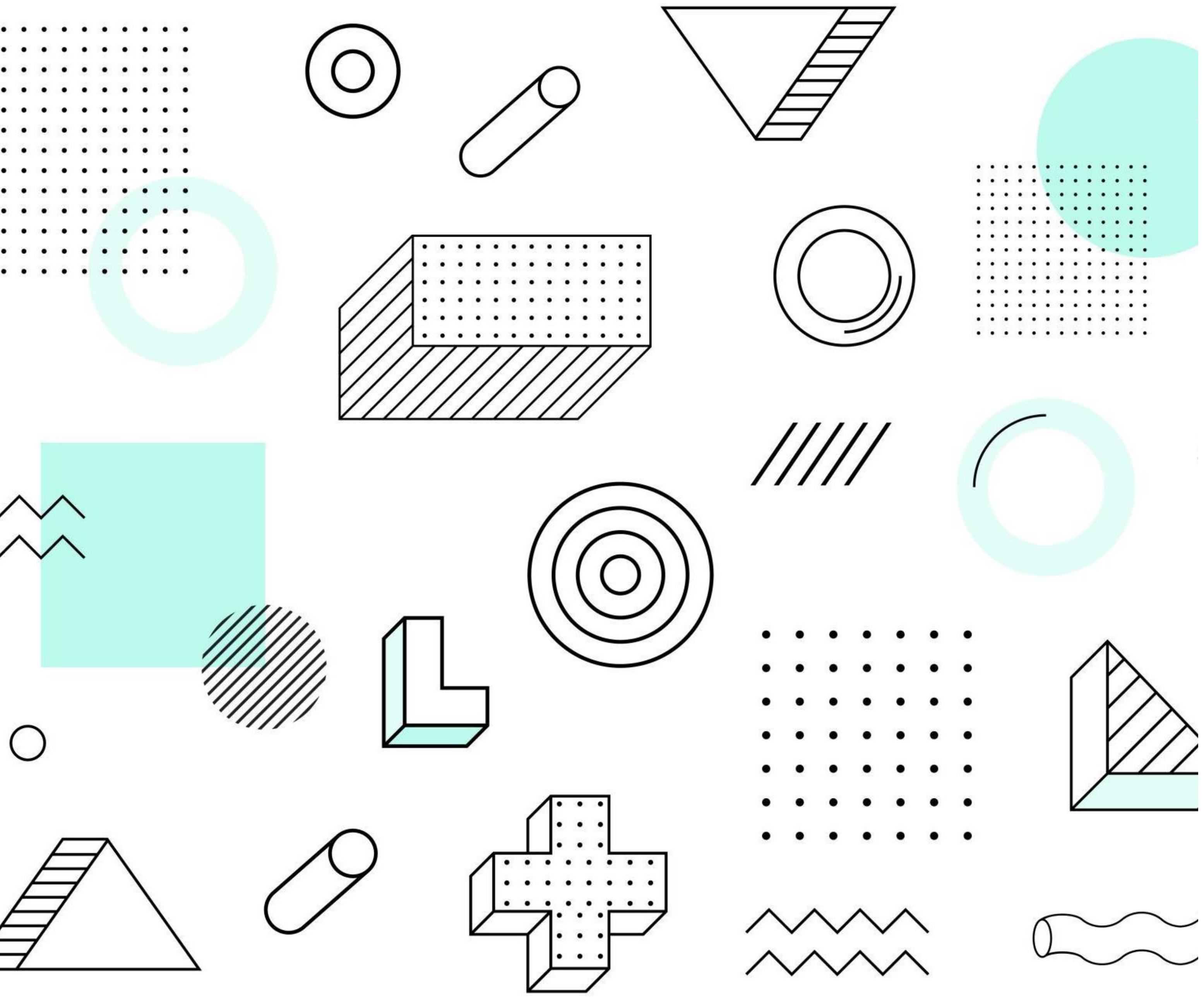

# STEMify Mzansi

Your pathway to exploring STEM careers

## Members of Quantum Forge

- Xadilam Mbambisa (developer)
- Buhle Magadla (developer)
- Asanda Mngqudaniso (project manager)
- Malibingwe Ngwilingwili (tester)

## Description

- STEMify Mzansi is a web app designed to help students explore different STEM career fields. Students take a quiz that classifies their interests into one of three categories: Industry, Academia, or Research Institute. Based on the results, the system provides personalized career recommendations.

## Tech stack

- **Python (Flask):**
  - Chosen for its simplicity and ease of implementation.
  - Flask is a lightweight Python framework capable of handling API calls and routing for the system.
- **HTML, CSS, JavaScript:**
  - Used to develop the frontend interface and static web pages.
  - Ensures interactivity and responsive design for users.
- **SQLite:**
  - Used as the database to store user data, quiz questions, career information, and progress tracking.
- **Replit:**
  - Initially used to host and test the web application when local hosting was not working.

## System architecture

- **Frontend:**
- Built with HTML, CSS, JavaScript, and Jinja templates.
- Components include: Home page, Quiz page, Dashboard, and Career information pages.
- **Backend:**
- Implemented using Flask.
- Handles routing and API endpoints.
- Manages authentication and session management using Flask-Login.
- Contains quiz logic for loading questions, scoring answers, and tracking progress.
- Implements the career recommendation engine to suggest careers based on quiz results.
- **Database:**
- Uses SQLite (development), along with JSON.
- Contains tables for Users, Quizzes, Careers, and User Progress.

## Implementation overview

- The STEMify Mzansi system is implemented as a full-stack web application using Python (Flask) for the backend, HTML/CSS/JavaScript with Jinja2 for the frontend, and SQLite for the database. The system allows students to explore STEM careers, take quizzes, and track progress.

- **1. Frontend Implementation**
- **Frameworks/Tools:**
  - Flask templates (Jinja2) for dynamic HTML
  - CSS
  - JavaScript for interactivity (quiz navigation, form validations)

.

- **Key Pages:**
  - **Home Page:** Welcome users and guide them to different sections
  - **Quiz Page:** Dynamically loads questions from the backend; allows submission
  - **Career Info Page:** Shows details of careers based on quiz results or chosen fields
  - **Dashboard:** Student and Advisor, Completed quizzes, recommended careers

- **Features:**
  - Responsive design for desktop and mobile
  - Form validation for user inputs
  - Dynamic content rendering from backend data

## 2. Backend Implementation

- **Framework:** Flask
- **Main Responsibilities:**
    - **Routing:** Handles URLs and directs to appropriate frontend templates
    - **Business Logic:**
        - Quiz management (load questions, check answers, calculate scores)
        - Career recommendation logic (based on quiz scores or selected interests)
        - User progress tracking

- **Authentication & Authorization:**
  - User registration and login via **Flask-Login**
  - Session management to track logged-in users
- **Database Interaction:**
  - Uses **Flask-SQLAlchemy ORM** to interact with SQLite
  - Models defined for Users, Quizzes, Careers, and Progress
  - Queries to fetch, insert, and update user and quiz data

- **3. Database Implementation**
- **Database:** SQLite (for development)
- **Tables/Models:**
  - **User:** `id`, `username`, `email`, `password_hash`, `progress`
  - **Quiz:** `id`, `question_text`, `options`, `correct_answer`
  - **Career:** `id`, `name`, `description`, `category`, `requirements`
  - **Progress:** `user_id`, `quiz_id`, `score`, `completion_date`
- **Features:**
  - Store structured JSON for quiz options and answers
  - Track user-specific data (completed quizzes, recommended careers)

- **4. Key Functionalities Implementation**
- **User Authentication:**
  - Secure password hashing
  - Login, logout, session management
- **Quiz Management:**
  - Dynamic question loading using JSON fields
  - Score calculation on submission
  - Store quiz results in the database
- **Career Recommendation Engine:**
  - Map quiz scores to career categories
  - Suggest careers dynamically based on user performance

## Testing

- **We used 2 main methods of testing the system:**
- **1. Usability Testing**
- Focused on user interaction with the system
- Checked ease of navigation and clarity of instructions
- Improved user experience and accessibility
- **2. Functional Testing**
- Verified that all features work correctly
- Tested registration, login, quizzes, scoring, and career recommendations
- Ensured backend and database function as expected

## Challenges

- During the development of STEMify Mzansi this semester, most challenges were related to **coding and testing the system**. Setting up a local development environment proved difficult due to the limited capabilities of the available PCs. Additionally, access to the Computer Science lab was restricted, further complicating testing and development.
- Initially, we used **Replit** to host and test the system, leveraging its built-in AI agent to identify and fix errors. However, Replit imposed limitations on agent usage, which made it difficult to conduct extensive testing.
- The issue of local hosting was eventually resolved. We discovered that many dependencies were outdated, which prevented the system from running correctly. After updating the dependencies, we were able to host the system locally and carry out all required testing successfully.

## Progress made

Significant progress has been made in the development of STEMify Mzansi. The **frontend interface** has been implemented, allowing users to navigate the system, view career information, and access the quiz. The **backend structure** using Flask and SQLAlchemy is in place, including user registration, login, and database integration for storing career data and user progress.

The quiz logic is functional. The student is able to login, take the quiz. After the quiz has been completed, it is then graded and the student is able to view their score and the best fit for them. From there they are able to view careers that match the best fit for their careers.

## Future plans

- Moving forward, the focus will be on enhancing the functionality and usability of STEMify Mzansi. Key plans include:
- **Fixing and Refining Quiz Logic:**
  - Ensure that the quiz accurately classifies students into Industry, Academia, or Research Institute based on their responses.
  - Implement a **robust scoring algorithm** and handle edge cases to improve reliability.
- **Expanding Career Information:**
  - Add more careers and detailed descriptions under each category.
  - Include information on required skills, educational pathways, and potential employers.
- **User Experience Improvements:**
  - Enhance the frontend interface for better navigation and interactivity.
  - Make the system mobile-friendly to increase accessibility.

- **Analytics and Progress Tracking:**
  - Implement features to track user quiz performance over time.
  - Provide personalized recommendations based on historical quiz results.
- **Deployment and Scalability:**
  - Transition from SQLite to PostgreSQL/MySQL for better performance in production.
  - Explore deployment options like Docker or cloud hosting for continuous access.
- **Additional Features (Optional):**
  - Add a feedback system for users to suggest improvements.
  - Introduce AI-driven career recommendations for more personalized guidance.