Advanced Algebra Visualizer

Complete Project Documentation

**Version: 1.0.0**Generated on: September 30, 2025  
Technology Stack: Streamlit, Python, SQLite, Plotly

# Table of Contents

1. Project Overview

2. System Architecture

3. Installation & Setup

4. Module Documentation

5. Database Schema

6. Features Guide

7. API Reference

8. Deployment Guide

9. Troubleshooting

10. Usage Examples

11. Future Enhancements

12. Support & Contact

# 1. Project Overview

**Advanced Algebra Visualizer** is an interactive web application built with Streamlit that provides comprehensive algebra learning tools with gamification, voice commands, and real-time visualization.

## Key Features

🔍 Interactive Equation Solvers (Quadratic, Polynomial)

📊 Real-time Graphing & Visualization

🎮 Gamification & Progress Tracking

🎤 Voice Command Interface

🔐 User Authentication System

🎨 Customizable Themes

🌍 Real-world Applications

🗃️ Database Management Tools

## Technology Stack

Frontend: Streamlit, Plotly

Backend: Python, SQLite

Math Engine: SymPy, NumPy

Voice Processing: SpeechRecognition, pyttsx3

Data Visualization: Plotly, Pandas

# 2. System Architecture

## File Structure

Advanced\_Algebra/  
├── app.py # Main application  
├── math\_engine.py # Mathematical computations   
├── gamification.py # Progress tracking & achievements  
├── auth.py # User authentication  
├── voice\_commands.py # Voice interface  
├── visualizations.py # Graph plotting  
├── theme.py # UI theming system  
├── config.py # Application configuration  
└── data/  
 └── user\_progress.db # SQLite database

## Data Flow

1. User Input → Streamlit Interface

2. Math Processing → Math Engine

3. Visualization → Plotly Graphs

4. Progress Tracking → Gamification Engine

5. Data Storage → SQLite Database

# 3. Installation & Setup

## Prerequisites

• Python 3.8+

• pip package manager

## Step-by-Step Installation

**1. Clone/Download the Project**

# Create project directory  
mkdir Advanced\_Algebra  
cd Advanced\_Algebra

**2. Create Virtual Environment (Recommended)**

python -m venv algebra\_env  
source algebra\_env/bin/activate # Linux/Mac  
algebra\_env\Scripts\activate # Windows

**3. Install Dependencies**

pip install streamlit plotly numpy pandas sympy speechrecognition pyttsx3

**4. Project Structure Setup**

# Create necessary directories  
mkdir -p data

**5. Run the Application**

streamlit run app.py

# 4. Module Documentation

## Main Application (app.py)

Purpose: Central application controller and UI manager

Key Components:

• AlgebraVisualizerApp class - Main application controller

• Navigation system with 6 main tabs

• Theme application and session management

Main Methods:

• render\_sidebar() - Control panel with user settings

• render\_quadratic\_solver() - Advanced equation solver

• render\_polynomial\_analyzer() - Polynomial analysis tools

• render\_identity\_prover() - Algebraic identity verification

• render\_real\_world\_apps() - Practical applications

• render\_database\_viewer() - Database monitoring

## Math Engine (math\_engine.py)

Purpose: Core mathematical computation engine

Key Features:

• Symbolic mathematics with SymPy

• Equation solving and simplification

• Derivative and integral calculations

• Identity verification

Main Methods:

• math\_engine.solve\_quadratic(a, b, c) - Solve quadratic equations

• math\_engine.expand\_expression(expr) - Expand algebraic expressions

• math\_engine.factor\_expression(expr) - Factor expressions

• math\_engine.calculate\_derivative(expr) - Compute derivatives

• math\_engine.prove\_identity(left, right) - Verify algebraic identities

# 5. Database Schema

## Users Table

CREATE TABLE users (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 username TEXT UNIQUE NOT NULL,  
 email TEXT UNIQUE NOT NULL,  
 password\_hash TEXT NOT NULL,  
 salt TEXT NOT NULL,  
 role TEXT DEFAULT 'student',  
 is\_verified INTEGER DEFAULT 1,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 last\_login TIMESTAMP,  
 profile\_data TEXT DEFAULT '{}'  
);

## User Progress Table

CREATE TABLE user\_progress (  
 user\_id TEXT PRIMARY KEY,  
 total\_points INTEGER DEFAULT 0,  
 current\_level INTEGER DEFAULT 1,  
 problems\_attempted INTEGER DEFAULT 0,  
 problems\_solved INTEGER DEFAULT 0,  
 streak\_days INTEGER DEFAULT 0,  
 last\_active DATE,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);

# 6. Features Guide

## Quadratic Equation Solver

Location: 🎯 Quadratic Solver tab

Features:

• Real-time equation solving

• Step-by-step solutions

• Graphical representation

• Discriminant analysis

• Vertex and axis of symmetry calculation

Usage:

1. Select equation format (Standard/Vertex/Factored)

2. Adjust coefficients using sliders

3. View step-by-step solution

4. Analyze graphical representation

## Polynomial Analyzer

Location: 📈 Polynomial Analyzer tab

Features:

• Polynomial input up to 6th degree

• Real and complex root finding

• Interactive graphing

• Root visualization on graph

Usage:

1. Select polynomial degree

2. Enter coefficients

3. Click 'Analyze Polynomial'

4. View roots and graph

# 7. API Reference

## Math Engine API

### solve\_quadratic(a, b, c)

Solves quadratic equations of form ax² + bx + c = 0

Parameters:

• a, b, c (float): Coefficients

Returns:

{  
 "type": "real|double|complex",  
 "roots": [root1, root2],  
 "discriminant": value  
}

# 8. Deployment Guide

## Local Deployment

1. Follow installation steps above

2. Run: streamlit run app.py

3. Access at: http://localhost:8501

## Cloud Deployment (Streamlit Sharing)

1. Prepare for Deployment

# requirements.txt  
streamlit==1.28.0  
plotly==5.15.0  
numpy==1.24.0  
pandas==2.0.0  
sympy==1.12  
speechrecognition==3.10.0  
pyttsx3==2.90

1. Push code to GitHub repository

2. Connect repository to Streamlit Sharing

3. Deploy automatically

# 9. Troubleshooting

## Common Issues

**Issue: Module Not Found Errors**

# Ensure all dependencies are installed  
pip install -r requirements.txt

**Issue: Database Connection Issues**

# Check database file exists  
import os  
if not os.path.exists('data'):  
 os.makedirs('data')

# 10. Usage Examples

## Basic Quadratic Solving

Using the math engine directly

# Using the math engine directly  
solution = math\_engine.solve\_quadratic(1, -3, 2)  
# Returns: x=1, x=2 (real roots)

## User Progress Tracking

Update progress after problem solving

# Update progress after problem solving  
user\_data, points = game\_engine.update\_user\_progress(  
 user\_id="user\_123",   
 concept="Quadratic Equations",   
 difficulty=2,   
 solved=True  
)

# 11. Future Enhancements

## Planned Features

• Mobile app version

• Collaborative problem solving

• Advanced calculus modules

• 3D graphing capabilities

• AI-powered problem generation

• Multi-language support

• Offline functionality

# 12. Support & Contact

## Getting Help

1. Check this documentation first

2. Review error messages in console

3. Search existing issues

4. Create new issue with details

## System Requirements

• Minimum: 2GB RAM, Python 3.8+

• Recommended: 4GB RAM, Python 3.9+

• Optimal: 8GB RAM, Python 3.11+

**🎉 Congratulations!** You now have complete documentation for the Advanced Algebra Visualizer project.