**File Explanation: app.py   
Core Architecture**

**1. Server Setup**

* **Flask Application**: Creates a web server with necessary routes and CORS support
* **Firebase Integration**: Uses Firestore for data storage with graceful fallback to mock implementation if credentials aren't available

**2. AI Generation Pipeline**

The application uses a tiered approach to website generation:

User Prompt → Try Gemini → Try Groq → Try Hugging Face → Fallback to Templates  
(*Each step has proper error handling to ensure users always get a result.)*

**3. Template System**

If AI services fail, the app uses built-in template generators for different website types:

* Ultra-modern templates with glassmorphism effects
* Portfolio templates
* Business templates
* Restaurant templates
* E-commerce templates

**Key Features**

**1. Multi-Model AI Strategy**

* **Primary**: Google's Gemini 1.5 Flash model
* **Secondary**: Groq's LLaMA3 model
* **Tertiary**: Hugging Face's CodeLLama
* **Backup**: Pre-built template system

**2. Project Management**

* Save generated websites to Firebase
* Retrieve user projects
* Download projects as HTML files
* Delete projects

**3. Smart Prompt Analysis**

* Extracts meaningful titles from prompts
* Determines website type based on prompt keywords
* Generates relevant tags automatically

**4. Modern UI Elements**

The templates include:

* Responsive design
* Glassmorphism effects
* Animated components
* Proper mobile support
* Modern CSS (Grid, Flexbox, Variables)

**Technical Implementation**

The code follows best practices including:

* Comprehensive error handling
* Logging for debugging
* API key management
* Service degradation strategy
* Clean separation of concerns

*The application demonstrates how to integrate multiple AI services with appropriate fallback mechanisms to create a reliable, production-ready web application.*

**File Explanation: migrate\_database.py   
Main Function**

migrate\_database()

* Entry point that initializes Firebase connection
* Orchestrates all migration steps
* Handles errors and provides status updates

**Three Key Migrations:**

**1.**migrate\_projects\_add\_fields()

Ensures all projects have standardized fields by adding:

* Empty tags array
* is\_public flag (defaulted to false)
* updated\_at timestamp
* version tracking (set to 1)

**2.**migrate\_user\_settings()

Updates user preferences structure:

* Creates default settings if missing (email notifications, dark mode, etc.)
* Adds new settings fields to existing users
* Sets profile completion status

**3.**create\_indexes()

Documents required database indexes for performance:

* Projects sorted by user and creation date
* Projects filtered by visibility
* Analytics data organization

**Technical Implementation**

* Uses Firebase Admin SDK with batch processing
* Provides detailed console logging with emoji status indicators
* Handles errors gracefully with try/except blocks

**File Explanation: setup\_database.py  
Main Function**

setup\_database()

* Initializes Firebase with service account credentials
* Creates core database collections with sample data
* Provides status feedback with emoji indicators

**Three Essential Collections Created:**

**1. Users Collection**

Stores user profiles with:

* Basic info (name, email)
* Account creation timestamp
* Personalized settings (notifications, UI preferences)

**2. Projects Collection**

Stores generated websites with:

* Project metadata (title, user ID)
* AI generation prompt
* Generated HTML code
* Timestamps for creation/updates
* Categorization tags
* Privacy settings

**3. Analytics Collection**

Tracks system usage with:

* User actions (website generation events)
* Timestamps
* Performance metrics (generation time, prompt length)
* Model information

**Technical Implementation**

* Uses Firebase Admin SDK for database operations
* Creates sample documents to establish expected data structure
* Handles errors with try/except pattern
* Provides clear console output about setup progress

**File Explanation: setup\_database.py  
Main Function**

setup\_database()

* Initializes Firebase with service account credentials
* Creates core database collections with sample data
* Provides status feedback with emoji indicators

**Three Essential Collections Created:**

**1. Users Collection**

Stores user profiles with:

* Basic info (name, email)
* Account creation timestamp
* Personalized settings (notifications, UI preferences)

**2. Projects Collection**

Stores generated websites with:

* Project metadata (title, user ID)
* AI generation prompt
* Generated HTML code
* Timestamps for creation/updates
* Categorization tags
* Privacy settings

**3. Analytics Collection**

Tracks system usage with:

* User actions (website generation events)
* Timestamps
* Performance metrics (generation time, prompt length)
* Model information

**Technical Implementation**

* Uses Firebase Admin SDK for database operations
* Creates sample documents to establish expected data structure
* Handles errors with try/except pattern
* Provides clear console output about setup progress

**File Explanation: auth.css, dashboard.css & style.css  
Overall Layout Structure**

**Overall Layout Structure**

* **Split-View Design**: Left sidebar navigation + main content area
* **Clean Visual Hierarchy**: White background with subtle shadows and borders
* **Consistent Color Scheme**: Indigo primary color (#4f46e5) with complementary grays

**Key Components**

**1. Navigation Sidebar (280px)**

* **Persistent Navigation**: Fixed-width sidebar with branded header
* **Visual Indicators**: Active state highlighting with blue accent border
* **User Profile Section**: Avatar display with name in footer area

**2. Project Management Interface**

* **Card-Based Grid**: Responsive layout adapting to screen size
* **Interactive Cards**: Hover effects with subtle elevation change
* **Project Metadata**: Organized display of project details and timestamps

**3. Content Creation Areas**

* **Focused Input Forms**: Clean white containers with proper spacing
* **Enhanced Form Elements**: Distinctive focus states with subtle highlights
* **Action Buttons**: Clear button hierarchy with primary/secondary styling

**4. Modal Windows**

* **Overlay System**: Fixed position with backdrop blur effect
* **Responsive Sizing**: Adapts to different screen sizes while maintaining readability
* **Project Preview**: Embedded iframe for website preview functionality

**5. Responsive Behavior**

* **Mobile Adaptation**: Switches to vertical layout on smaller screens (below 768px)
* **Simplified Navigation**: Full-width sidebar becomes horizontal on mobile
* **Adjusted Spacing**: Reduced padding and simplified grids for smaller devices

**File Explanation: auth.js, dashboard.js & script.js  
Core Functionality**

* **Firebase Integration**: Initializes Firebase authentication and Firestore database
* **Form Management**: Handles switching between login and signup views
* **Validation Logic**: Enforces security rules (password matching, minimum length)
* **Visual Feedback**: Disables buttons during processing and displays status

**Authentication Methods**

* **Email/Password Authentication**: Traditional signup and login flows
* **Google Authentication**: Single-click social login option
* **Session Management**: Automatic redirect to dashboard for logged-in users

**User Data Management**

* **Profile Creation**: Stores user information in Firestore database
* **Default Settings**: Initializes user preferences for the application
* **Secure Redirects**: Routes users to appropriate pages based on auth state

**Implementation Details**

* **Real-time Feedback**: Console logging with emoji indicators for debugging
* **Error Handling**: Comprehensive try/catch blocks with user-friendly messages
* **State Persistence**: Auth state observer detects existing sessions

**File Explanation: auth.html, dashboard.html & index.html  
User Interface Structure**

**1. Navigation Header**

* Brand identity with layered logo design
* Dynamic authentication controls (sign in/up or dashboard/logout)
* Clean, accessible button styling with icon integration

**2. Hero Marketing Section**

* Eye-catching gradient text highlighting
* Concise value proposition explaining the service
* Trust indicators with metrics (50K+ websites, generation time)

**3. AI Chat Interface**

* Natural conversation format with assistant/user messaging
* Accessible text input with keyboard shortcuts
* Visual authentication gate for non-logged-in users

**4. Example Templates Gallery**

* Six pre-configured website suggestions
* Clear categorization (portfolio, landing page, restaurant, etc.)
* One-click prompt population for faster starts

**5. Live Preview System**

* Interactive device toggle (desktop/tablet/mobile views)
* Full-sized iframe preview environment
* Action buttons for downloading or saving projects

**6. Loading Experience**

* Multi-stage progress visualization
* Animated spinner with concentric rings
* Step indicators showing generation phases

**Technical Integration**

* Firebase authentication and storage connectivity
* Font Awesome icon library for visual elements
* Responsive design patterns for all screen sizes

**File Explanation: auth.html, dashboard.html & index.html  
Web Framework**

* **Flask (2.3.3)**: Lightweight web server framework handling API requests
* **Werkzeug (2.3.7)**: WSGI utilities supporting Flask's core functionality

**API Infrastructure**

* **Flask-CORS (4.0.0)**: Enables secure cross-origin requests from frontend
* **Requests (2.31.0)**: HTTP client for external API communication (AI services)

**Database & Authentication**

* **Firebase-Admin (6.2.0)**: Server-side SDK for Firebase database and auth

**Configuration & Deployment**

* **Python-dotenv (1.0.0)**: Environment variable management for security
* **Gunicorn (21.2.0)**: Production-grade WSGI server for reliable hosting