**ODAVILLE WEBSITE AND ADMIN PANEL DOCUMENTATION**

**Table of Contents**

1. [Project Overview](#project-overview)
2. [Access Credentials](#access-credentials)
3. [System Architecture](#system-architecture)
4. [Repository Structure](#repository-structure)
5. [Deployment Process](#deployment-process)
6. [Development Workflow](#development-workflow)
7. [Content Management System](#content-management-system)
8. [Technical Implementation Details](#technical-implementation-details)
9. [Troubleshooting Guide](#troubleshooting-guide)
10. [Future Enhancements](#future-enhancements)
11. [Maintenance Procedures](#maintenance-procedures)
12. [Appendix: API Reference](#appendix-api-reference)

**1. Project Overview**

The Odaville platform consists of two main components:

**Main Website**: A public-facing website showcasing Odaville's architectural products including windows, doors, and other architectural elements. The website features a responsive design and content managed through a custom CMS.

**Admin Panel**: A secure administration interface for content management including gallery items, blog posts, and product listings.

**Key Features**

* Responsive frontend design
* Content management system
* Image upload and management
* Blog post creation and editing
* Product catalog management
* Gallery management

**Technologies Used**

* **Frontend**: HTML5, CSS3, JavaScript
* **Backend**: Node.js, Serverless Functions
* **Database**: MongoDB
* **Storage**: AWS S3
* **Deployment**: Vercel
* **Version Control**: Git/GitHub

**2. Access Credentials**

**Application Access**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **System** | **URL** | **Username** | **Password** | | --- | --- | --- | --- | | Main Website | <https://www.odaville.com> | N/A | N/A | | Admin Panel | <https://admin.odaville.com> | [ADMIN\_USERNAME] | [ADMIN\_PASSWORD] | | Development Site | <https://odaville-dev.vercel.app> | N/A | N/A | | Development Admin | <https://odaville-admin-dev.vercel.app> | [DEV\_ADMIN\_USERNAME] | [DEV\_ADMIN\_PASSWORD] | |

**Repository Access**

| **Repository** | **URL** | **Access Level** |
| --- | --- | --- |
| Main Website | [https://github.com/[ORGANIZATION]/odaville](https://github.com/%5BORGANIZATION%5D/odaville) | [public] |
| Admin Panel | [https://github.com/[ORGANIZATION]/odaville-admin](https://github.com/%5BORGANIZATION%5D/odaville-admin) | [admin-only] |

**Service Credentials**

| **Service** | **Account** | **Details** |
| --- | --- | --- |
| Vercel | [VERCEL\_ACCOUNT] | Project names: "odaville" and "odaville-admin" |
| MongoDB | [MONGODB\_ACCOUNT] | Database name: "odaville" |
| AWS | [AWS\_ACCOUNT] | S3 Bucket: "odaville-uploads " |

**3. System Architecture**

The Odaville platform follows a JAMstack architecture with serverless functions:

![System Architecture Diagram]

**Components**

1. **Front-end Client**
   * Static HTML, CSS, and JavaScript
   * Responsive design for multiple device sizes
   * Client-side content loading with API connection
2. **Serverless Functions**
   * Authentication and authorization
   * Content management APIs
   * Image upload processing
3. **MongoDB Database**
   * Content storage (blog posts, gallery items, products)
   * User authentication data
4. **AWS S3 Storage**
   * Media file storage (images, documents)
   * Content delivery network for fast loading
5. **Vercel Deployment**
   * Continuous integration and deployment
   * Edge network for global content delivery
   * Environment configuration management

**4. Repository Structure**

**Main Website Repository Structure**

odaville/

├── public/ # Static assets directory

│ ├── index.html # Main entry point

│ ├── about.html # About page

│ ├── blog.html # Blog listing page

│ ├── blog-detail.html # Blog detail page

│ ├── gallery.html # Gallery page

│ ├── products.html # Products page

│ ├── contact.html # Contact page

│ ├── css/ # Stylesheet directory

│ │ ├── styles.css # Main stylesheet

│ │ ├── blog-styles.css # Blog-specific styles

│ │ └── gallery-styles.css # Gallery-specific styles

│ ├── js/ # JavaScript directory

│ │ ├── script.js # Main script file

│ │ ├── db-loader.js # Data loading script

│ │ ├── products.js # Products functionality

│ │ └── navbar.js # Navigation functionality

│ ├── images/ # Image assets

│ │ ├── gallery/ # Gallery images

│ │ ├── blog/ # Blog images

│ │ └── products/ # Product images

│ └── videos/ # Video assets

├── vercel.json # Vercel deployment configuration

└── README.md # Project documentation

**Admin Panel Repository Structure**

odaville-admin/

├── public/ # Static assets directory

│ ├── admin/ # Admin interface files

│ │ ├── login.html # Login page

│ │ ├── dashboard.html # Admin dashboard

│ │ ├── blog-manage.html # Blog management

│ │ ├── gallery-manage.html # Gallery management

│ │ ├── product-manage.html # Product management

│ │ ├── css/ # Admin stylesheets

│ │ └── js/ # Admin scripts

│ │ ├── login.js # Authentication script

│ │ ├── dashboard.js # Dashboard script

│ │ └── admin.js # General admin script

│ └── index.html # Redirect to admin

├── api/ # API endpoints

│ ├── auth.js # Authentication API

│ ├── blog.js # Blog management API

│ ├── gallery.js # Gallery management API

│ ├── products.js # Products management API

│ ├── utils.js # Utility functions

│ ├── models.js # Database models

│ └── verify.js # Token verification

├── vercel.json # Vercel deployment configuration

└── README.md # Project documentation

**5. Deployment Process**

**Main Website Deployment**

The main website is deployed on Vercel with the following process:

1. **Code Push**: Changes are pushed to the main branch on GitHub
2. **Build Process**: Vercel automatically detects changes and starts the build process
3. **Deployment**: Built files are deployed to Vercel's global CDN
4. **Domain Configuration**: The site is available at [www.odaville.com](http://www.odaville.com)

**Admin Panel Deployment**

1. **Code Push**: Changes are pushed to the main branch on GitHub
2. **Build Process**: Vercel automatically builds the admin panel
3. **Deployment**: Admin panel is deployed as a separate Vercel project
4. **Domain Configuration**: Admin panel is available at admin.odaville.com

**Configuration Settings**

Key environment variables are set in the Vercel project settings:

| **Variable** | **Purpose** |
| --- | --- |
| MONGODB\_URI | MongoDB connection string |
| JWT\_SECRET | Secret key for JWT token generation |
| AWS\_ACCESS\_KEY\_ID | AWS access credentials |
| AWS\_SECRET\_ACCESS\_KEY | AWS secret key |
| AWS\_REGION | AWS region for S3 bucket |
| AWS\_BUCKET\_NAME | S3 bucket name for file storage |

**6. Development Workflow**

**Git Workflow**

1. **Main Branch**: Production-ready code
2. **Development Branch**: Integration branch for new features
3. **Feature Branches**: Created for specific features or fixes
   * Format: feature/feature-name or fix/issue-description
4. **Pull Request Process**:
   * Create PR from feature branch to development branch
   * Review code and approve changes
   * Merge to development
5. **Deployment Process**:
   * Merge development to main for production deployment
   * Vercel automatically deploys changes

**Local Development Setup**

1. **Clone Repositories**:

bash

git clone https://github.com/[ORGANIZATION]/odaville.git

git clone https://github.com/[ORGANIZATION]/odaville-admin.git

1. **Install Dependencies**:

bash

*# No npm dependencies for main website*

cd odaville-admin

npm install

1. **Set Up Environment Variables**:
   * Create a .env file in the odaville-admin directory
   * Add the following variables:
   * MONGODB\_URI=”Check Environment Variable Documents for these credentials”
   * JWT\_SECRET=”Check Environment Variable Documents for these credentials”
   * AWS\_ACCESS\_KEY\_ID=”Check Environment Variable Documents for these credentials”
   * AWS\_SECRET\_ACCESS\_KEY=”Check Environment Variable Documents for these credentials”
   * AWS\_REGION=”Check Environment Variable Documents for these credentials”

AWS\_BUCKET\_NAME=”Check Environment Variable Documents for these credentials”

1. **Run Development Servers**:
   * For main website, use a local HTTP server:

bash

cd odaville

npx serve public

* + For admin panel:

bash

cd odaville-admin

vercel dev

**7. Content Management System**

**Authentication**

1. **Login Process**:
   * Navigate to admin.odaville.com
   * Enter credentials (username and password)
   * System generates JWT token for authentication
   * Token is stored in browser localStorage/sessionStorage
2. **Session Management**:
   * Sessions last for 24 hours by default
   * "Remember Me" option extends session using localStorage

**Content Types**

1. **Blog Posts**:
   * Title, content, author, category, image, published status
   * Rich text editor for content creation
   * Image upload capability
   * Publish/unpublish toggle
2. **Gallery Items**:
   * Title, description, category, image, featured status
   * Category filtering
   * Featured items appear larger in the gallery grid
3. **Products**:
   * Title, subtitle, description, category, image, featured status
   * Category filtering
   * Order control for display sequence

**Image Management**

1. **Upload Process**:
   * Select image in the admin interface
   * Image is uploaded to AWS S3
   * URL is stored in the database
   * Image is displayed on the website
2. **Supported Formats**:
   * JPG/JPEG
   * PNG
   * WebP
   * GIF

**Content Workflow**

1. **Creating Content**:
   * Log in to admin panel
   * Navigate to appropriate section (Blog, Gallery)
   * Click "Add New" button
   * Fill in required fields
   * Upload image if needed
   * Save or publish
2. **Editing Content**:
   * Find content in listing page
   * Click edit button
   * Make changes
   * Save changes
3. **Deleting Content**:
   * Find content in listing page
   * Click delete button
   * Confirm deletion
   * System removes content and associated images

**8. Technical Implementation Details**

**Frontend Implementation**

1. **Data Loading**:
   * Data is loaded through the db-loader.js script
   * API endpoints are called to fetch content
   * Content is rendered dynamically on the page
   * Mock data fallback when API is unavailable
2. **Responsive Design**:
   * Mobile-first approach
   * Breakpoints for different device sizes
   * Flexbox and CSS Grid for layouts
3. **JavaScript Features**:
   * Gallery filtering
   * Blog pagination
   * Navigation handling
   * Smooth scrolling
   * Animations on scroll

**Backend Implementation**

1. **API Endpoints**:
   * RESTful API design
   * JWT authentication
   * MongoDB connection
   * S3 integration for file uploads
2. **Authentication Flow**:
   * Username/password validation
   * JWT token generation
   * Token verification middleware
   * Role-based access control
3. **Database Models**:
   * User model for authentication
   * Blog model for blog posts
   * Gallery model for gallery items
   * Product model for products

**Data Flow**

1. **Read Operations**:
   * Frontend requests data from API
   * API queries MongoDB
   * Data is returned to frontend
   * Frontend renders data
2. **Write Operations**:
   * Admin submits form data
   * API receives and validates data
   * Data is saved to MongoDB
   * Confirmation is returned to admin panel

**9. Troubleshooting Guide**

**Common Issues and Solutions**

1. **API Connection Errors**:
   * **Symptom**: Website shows "Failed to load" messages
   * **Solution**:
     + Check MongoDB connection string
     + Verify API endpoints in vercel.json
     + Check network connectivity
     + Mock data should display as fallback
2. **Image Upload Failures**:
   * **Symptom**: Images fail to upload in admin panel
   * **Solution**:
     + Verify AWS credentials
     + Check S3 bucket permissions
     + Ensure correct CORS configuration
     + Check file size limits
3. **Authentication Issues**:
   * **Symptom**: Unable to log in to admin panel
   * **Solution**:
     + Verify credentials
     + Check JWT\_SECRET environment variable
     + Clear browser cache and cookies
     + Look for console errors
4. **Deployment Failures**:
   * **Symptom**: Vercel deployment fails
   * **Solution**:
     + Check vercel.json configuration
     + Look for "Mixed routing properties" error
     + Verify build settings
     + Check environment variables

**Debugging Tools**

1. **Browser Developer Tools**:
   * Console for JavaScript errors
   * Network tab for API requests
   * Application tab for localStorage/cookies
2. **Vercel Logs**:
   * Function execution logs
   * Build logs for deployment issues
   * Environment variable verification
3. **MongoDB Atlas Dashboard**:
   * Database connection status
   * Query monitoring
   * Performance metrics

**10. Future Enhancements**

**Planned Features**

1. **User Management**:
   * Multiple admin user accounts
   * Role-based permissions
   * User activity logging
2. **Advanced Content Management**:
   * Content scheduling
   * Custom content types
   * Version history and rollback
3. **Performance Improvements**:
   * Image optimization pipeline
   * Advanced caching
   * Code splitting and lazy loading
4. **Analytics Integration**:
   * Google Analytics integration
   * Content performance metrics
   * User behavior tracking

**Architectural Improvements**

1. **API Enhancement**:
   * GraphQL API implementation
   * Real-time updates with WebSockets
   * Enhanced error handling
2. **Security Enhancements**:
   * Two-factor authentication
   * Enhanced CORS policies
   * Regular security audits

**11. Maintenance Procedures**

**Regular Maintenance Tasks**

1. **Database Backups**:
   * MongoDB Atlas automatic backups
   * Schedule: Daily
   * Retention: 7 days
2. **Content Audits**:
   * Review and update outdated content
   * Schedule: Monthly
   * Responsibility: Content manager
3. **Security Updates**:
   * Update dependencies
   * Review access logs
   * Schedule: Monthly
   * Responsibility: Developer

**Emergency Procedures**

1. **Site Outage**:
   * Check Vercel status
   * Verify MongoDB connection
   * Review deployment logs
   * Rollback to last working deployment if needed
2. **Security Breach**:
   * Reset all passwords
   * Revoke and reissue JWT secret
   * Audit access logs
   * Review code for vulnerabilities

**12. Appendix: API Reference**

**Authentication Endpoints**

* **POST /api/auth/login**
  + Description: Authenticates user and returns JWT token
  + Request Body: { username, password }
  + Response: { success, token, user }
* **GET /api/auth/verify**
  + Description: Verifies JWT token
  + Headers: Authorization: Bearer {token}
  + Response: { authenticated, user }

**Content Endpoints**

* **GET /api/blog**
  + Description: Retrieves blog posts
  + Query Parameters: published, category, limit
  + Response: Array of blog posts
* **GET /api/gallery**
  + Description: Retrieves gallery items
  + Query Parameters: category, featured
  + Response: Array of gallery items
  + Response: Array of products

**Content Management Endpoints**

* **POST /api/blog**
  + Description: Creates new blog post
  + Headers: Authorization: Bearer {token}
  + Request Body: Form data with blog fields and image
  + Response: Created blog post
* **POST /api/gallery**
  + Description: Creates new gallery item
  + Headers: Authorization: Bearer {token}
  + Request Body: Form data with gallery fields and image
  + Response: Created gallery item

*This documentation was last updated on May 2, 2025.*