**Fusion Electronics - E-commerce Application Documentation**

**📋 Project Overview**

The **Fusion Electronics E-commerce Application** is a comprehensive full-stack MERN (MongoDB, Express.js, React.js, Node.js) web application designed to provide a modern online shopping experience for electronics products. The system features AI-powered product recommendations using vector databases (Pinecone, Weaviate, FAISS), user authentication with JWT, complete shopping cart functionality, and comprehensive order management with real-time tracking.

**🎯 Use Cases**

**1. User Authentication & Authorization**

* **Registration System**: Secure user account creation with email validation and password hashing (BCrypt - 10 salt rounds)
* **Login System**: JWT-based authentication with token expiration (48 hours)
* **Password Management**: Forgot password and reset password functionality with email verification
* **User Profile**: View and manage personal information and order history
* **Role-Based Access**: Support for Guest Users, Registered Customers, and Admin roles

**2. Product Catalog Management**

* **Product Browsing**: View comprehensive product listings with images, descriptions, and pricing
* **Product Details**: Detailed product information including specifications, ratings, reviews, and stock availability
* **Category Filtering**: Organize and filter products by categories (Electronics, Computers, Gaming, Audio, etc.)
* **Product Search**: Real-time search functionality with debounced input for optimized performance
* **Product Ratings**: User-generated ratings and reviews system

**3. AI-Powered Product Recommendations**

* **Vector-Based Similarity**: Uses Pinecone (primary), Weaviate, and FAISS for semantic product matching
* **Similar Products**: Display 5 similar products on product detail pages based on vector embeddings
* **Personalized Recommendations**: Generate recommendations based on user browsing history and cart items
* **Heuristic Fallback**: Metadata-based scoring when vector search is unavailable (category, brand, price affinity)
* **Embedding Generation**: Google Generative AI (text-embedding-004) for 768-dimensional product vectors

**4. Shopping Cart & Checkout**

* **Cart Management**: Add, remove, and update product quantities in shopping cart
* **Cart Persistence**: Local storage integration for cart state preservation across sessions
* **Checkout Process**:
  + Comprehensive form with billing, shipping, and payment information
  + Client-side credit card validation using Luhn algorithm
  + Automatic card type detection (Visa, Mastercard, Amex, Discover, Diners Club, JCB)
  + Real-time validation with visual error feedback
  + Expiry date and CVC validation
* **Order Creation**: Backend validation and order processing with unique order number generation

**5. Order Management & Tracking**

* **Order Confirmation**: Post-purchase confirmation with order details and estimated delivery
* **Order Tracking**: Real-time order status tracking using order number and email
* **Order History**: View past orders with detailed information for registered users
* **Status Updates**: Track orders through various stages (Processing, Shipped, In Transit, Delivered)

**6. Customer Support & Information**

* **Support Page**: Contact information, FAQ section, and inquiry form
* **Terms of Service**: Comprehensive terms and conditions
* **Privacy Policy**: Detailed privacy policy and data protection information
* **Shipping & Returns**: Shipping policies, delivery information, and return procedures
* **Newsletter Subscription**: Email subscription for updates and promotions

**7. API Services**

* **RESTful APIs**: Comprehensive endpoints for all system operations
* **Swagger Documentation**: Interactive API documentation at /api-docs
* **OpenAPI Specification**: Complete API specification in openapi.yaml
* **Error Handling**: Industry-standard HTTP status codes with detailed error messages
* **Rate Limiting**: Protection against API abuse (429 Too Many Requests)

**8. Search & Discovery**

* **Text Search**: Case-insensitive regex search on product names and descriptions
* **Search Suggestions**: Real-time search results as user types
* **Filter & Sort**: Advanced filtering and sorting capabilities
* **Pagination**: Paginated search results for better performance

**🖥️ User Interface (UI) Components**

**1. Authentication Pages**

* **Login Page** (src/pages/Login.jsx):
  + Clean, modern Material-UI design with form validation
  + JWT token-based authentication
  + Error handling with user-friendly notifications
  + Responsive design for mobile and desktop
* **Register Page** (src/pages/Register.jsx):
  + User account creation with validation
  + Password strength requirements (minimum 6 characters)
  + Email format validation
  + Real-time error feedback
* **Forgot Password Page** (src/pages/ForgotPassword.jsx):
  + Email-based password reset initiation
  + Validation and error handling
* **Reset Password Page** (src/pages/ResetPassword.jsx):
  + Secure password reset with token verification
  + Password confirmation matching

**2. Main Navigation**

* **Navigation Bar** (src/components/NavigationBar.jsx):
  + Responsive top navigation with logo
  + Search bar with real-time search
  + Shopping cart badge with item count
  + User authentication status display
  + Mobile-friendly hamburger menu
* **Home Page** (src/pages/Home.jsx):
  + Hero section with featured banners
  + Featured products carousel
  + AI-powered recommended products section
  + Category quick links
  + Promotional sections

**3. Product Catalog Interface**

* **Shop Page** (src/pages/Shop.jsx):
  + Grid layout of all products
  + Category filtering sidebar
  + Sort options (price, rating, popularity)
  + Pagination controls
  + Quick "Add to Cart" buttons
* **Product Details Page** (src/pages/ProductDetails.jsx):
  + Large product images with zoom capability
  + Comprehensive product information
  + Star rating display
  + Stock availability indicator
  + Add to cart functionality
  + Similar products section (AI-powered)
  + Product specifications and description
* **Product Card Component** (src/components/ProductCard.jsx):
  + Reusable product tile component
  + Product image, name, price, rating
  + Quick action buttons
  + Hover effects and animations

**4. Shopping Experience**

* **Cart Page** (src/pages/Cart.jsx):
  + Detailed cart item listing
  + Quantity adjustment controls
  + Remove item functionality
  + Subtotal and total calculations
  + Tax and shipping estimates
  + Proceed to checkout button
  + Empty cart state handling
* **Checkout Page** (src/pages/Checkout.jsx):
  + Multi-section checkout form
  + Order summary sidebar
  + Billing information form
  + Shipping address form
  + Credit card payment form with visual card display
  + Real-time form validation
  + Terms and conditions acceptance
* **Checkout Form Component** (src/components/CheckoutForm.jsx):
  + Interactive credit card visualization using react-credit-cards-2
  + Real-time card number formatting
  + Luhn algorithm validation
  + Expiry date and CVC validation
  + Card type auto-detection with icon display

**5. Post-Purchase Pages**

* **Order Success Page** (src/pages/OrderSuccess.jsx):
  + Order confirmation message
  + Order number display
  + Order summary with items
  + Estimated delivery date
  + Next steps and tracking information
* **Order Tracking Page** (src/pages/OrderTracking.jsx):
  + Order lookup form (order number + email)
  + Order status timeline
  + Shipment tracking details
  + Delivery address confirmation
  + Order history display

**6. Information Pages**

* **About Page** (src/pages/About.jsx):
  + Company information and mission
  + Team introduction
  + Technology stack showcase
* **Support Page** (src/pages/Support.jsx):
  + Contact form for customer inquiries
  + FAQ accordion
  + Support channels (email, phone)
  + Business hours information
* **Privacy Policy** (src/pages/Privacy.jsx):
  + Data collection and usage policies
  + Cookie policy
  + User rights and data protection
* **Terms of Service** (src/pages/Terms.jsx):
  + Terms and conditions
  + User responsibilities
  + Limitation of liability
* **Shipping & Returns** (src/pages/ShippingReturns.jsx):
  + Shipping methods and costs
  + Delivery timeframes
  + Return policy and procedures
  + Refund information

**7. Responsive Design Features**

* **CSS Styling** (src/App.css):
  + Modern gradient designs and color schemes
  + Mobile-first responsive layout
  + Consistent typography and spacing
  + Material-UI theme customization
  + Interactive buttons with hover effects
  + Loading states and animations
* **Footer Component** (src/components/Footer.jsx):
  + Multi-column footer layout
  + Quick links to important pages
  + Newsletter subscription form
  + Social media links
  + Copyright and legal information
* **Scroll to Top** (src/components/ScrollToTop.jsx):
  + Automatic scroll to top on route changes
  + Smooth scrolling behavior

**8. Notification System**

* **Notification Provider** (src/context/NotificationProvider.jsx):
  + Toast-style notifications
  + Success, error, warning, info message types
  + Auto-hide functionality
  + Queue management for multiple notifications

**🛠️ Technology Stack**

**Backend Technologies**

| **Category** | **Technology** | **Version / Description** |
| --- | --- | --- |
| **Core Framework** | Node.js | 18.x (JavaScript runtime) |
|  | Express.js | 4.x (Web application framework) |
| **Database** | MongoDB | 6.x (NoSQL document database) |
|  | Mongoose | ODM for MongoDB with schema validation |
| **Authentication** | jsonwebtoken (JWT) | Token-based authentication |
|  | bcryptjs | Password hashing and encryption |
| **Vector Databases** | Pinecone | Serverless vector database (primary) |
|  | Weaviate | Optional vector database |
|  | FAISS (faiss-node) | Facebook AI Similarity Search |
| **AI/ML Integration** | Google Generative AI | text-embedding-004 for embeddings |
|  | LangChain | LLM framework for vector operations |
| **API Documentation** | Swagger UI Express | Interactive API documentation |
|  | swagger-jsdoc | JSDoc to Swagger specification |
| **Validation** | express-validator | Request validation middleware |
| **HTTP Client** | Axios | Promise-based HTTP client |
| **CORS** | cors | Cross-Origin Resource Sharing |
| **Environment** | dotenv | Environment variable management |

**Frontend Technologies**

| **Category** | **Technology** | **Description** |
| --- | --- | --- |
| **Core Framework** | React.js | 18.x (UI library) |
|  | React Router DOM | 6.x (Client-side routing) |
| **UI Library** | Material-UI (MUI) | 5.x (Component library) |
|  | @mui/icons-material | Material Design icons |
| **State Management** | React Context API | Global state management |
|  | React Hooks | useState, useEffect, useContext, useCallback |
| **HTTP Client** | Axios | API requests with retry logic |
| **Form Handling** | React Hook Form | Efficient form validation (optional) |
| **Credit Cards** | react-credit-cards-2 | Credit card visualization |
| **Notifications** | Custom NotificationProvider | Toast-style notifications |

**Build & Development Tools**

| **Category** | **Technology** | **Description** |
| --- | --- | --- |
| **Build System** | npm | Node Package Manager |
|  | CRACO | Create React App Configuration Override |
|  | Webpack | Module bundler (via CRA) |
|  | Babel | JavaScript transpiler |
| **Testing Framework** | Jest | Unit and integration testing |
|  | React Testing Library | React component testing |
|  | Supertest | HTTP assertion library for backend |
| **Code Quality** | ESLint | JavaScript linting |
|  | Prettier | Code formatting |
| **Development Tools** | Nodemon | Backend hot-reloading |
|  | React DevTools | React debugging |
|  | MongoDB Compass | Database GUI |

**DevOps & CI/CD**

| **Category** | **Technology** | **Description** |
| --- | --- | --- |
| **Version Control** | Git | Distributed version control |
|  | GitHub | Repository hosting and collaboration |
| **CI/CD Pipeline** | GitHub Actions | Automated build, test, and deployment |
|  | Workflow Configuration | .github/workflows/ci.yml |
| **Containerization** | Docker | Container platform |
|  | Docker Compose | Multi-container orchestration |
| **Orchestration** | Kubernetes | Container orchestration |
|  | kubectl | Kubernetes CLI |
| **Cloud Deployment** | Vercel | Frontend hosting (Primary) |
|  | Render | Backend hosting (Backup) |
|  | MongoDB Atlas | Cloud database hosting |
| **Infrastructure as Code** | Terraform | Infrastructure provisioning |
|  | Packer | Machine image creation |
|  | Nomad | Workload orchestration |
| **Security** | HashiCorp Vault | Secrets management |

**Development Environment**

* **IDEs**: Visual Studio Code, JetBrains WebStorm
* **Runtime**: Node.js 18.x, npm 9.x
* **API Testing**: Postman, Swagger UI
* **Database Tools**: MongoDB Compass, Robo 3T
* **Utilities**:
  + http-proxy-middleware (Development proxy)
  + cross-env (Cross-platform environment variables)
  + concurrently (Run multiple commands)

**📊 System Architecture**

**Application Layers**

1. **Presentation Layer**:
   * React components and pages
   * Material-UI design system
   * Client-side routing with React Router
   * Responsive layouts for mobile and desktop
2. **API Layer**:
   * Express.js REST controllers
   * Route handlers for all endpoints
   * Request validation middleware
   * Error handling middleware
   * CORS configuration
3. **Service Layer**:
   * Business logic and transaction management
   * Pinecone synchronization service
   * Weaviate integration service
   * Embedding generation service
   * Authentication service
4. **Data Access Layer**:
   * Mongoose schemas and models
   * MongoDB CRUD operations
   * Database connection management
   * Query optimization
5. **Vector Database Layer**:
   * Pinecone client for similarity search
   * Weaviate client for optional queries
   * FAISS index for local similarity search
   * Automatic synchronization on data changes
6. **Security Layer**:
   * JWT authentication middleware
   * Password encryption with BCrypt
   * Input validation and sanitization
   * Rate limiting and CORS policies

**Design Patterns**

* **MVC (Model-View-Controller)**: Separation of concerns in backend
* **Component-Based Architecture**: Reusable React components
* **Repository Pattern**: Data access abstraction with Mongoose
* **Dependency Injection**: Loose coupling with Express middleware
* **Observer Pattern**: React Context for state management
* **Singleton Pattern**: Database connection and API clients
* **Factory Pattern**: Product and order creation
* **Strategy Pattern**: Recommendation algorithms (vector vs heuristic)

**Data Flow**

User Interface (React)

↓ HTTP Requests

API Gateway (Express)

↓ Route Handling

Controllers

↓ Business Logic

Services

↓ Data Operations

MongoDB ← → Pinecone/Weaviate

**🔒 Security Features**

**Authentication**

* **JWT Token-Based**: Stateless authentication with 48-hour token expiration
* **Password Encryption**: BCrypt hashing with 10 salt rounds
* **Session Management**: Secure token storage and validation
* **Password Reset**: Email-based secure password reset flow

**Authorization**

* **Role-Based Access Control (RBAC)**: Guest, Customer, Admin roles
* **Protected Routes**: Frontend route guards for authenticated pages
* **API Endpoint Protection**: JWT verification middleware
* **Method-Level Security**: Fine-grained access control

**Input Validation**

* **Client-Side**: Form validation with React Hook Form
* **Server-Side**: Express-validator for request validation
* **Sanitization**: XSS prevention and input cleaning
* **Type Checking**: Mongoose schema validation

**Security Headers**

* **CORS**: Configured allowed origins
* **CSRF Protection**: Token-based CSRF prevention
* **Secure Cookies**: HttpOnly and Secure flags
* **Content Security Policy**: XSS protection headers

**Data Protection**

* **Environment Variables**: Sensitive data in .env files (not committed)
* **API Key Management**: Secure storage of API keys
* **Database Security**: MongoDB connection string encryption
* **Rate Limiting**: API abuse prevention

**📈 Monitoring & Observability**

**Application Monitoring**

* **Health Checks**: Backend health endpoint at /api/health
* **Database Status**: MongoDB connection monitoring
* **API Performance**: Response time tracking
* **Error Tracking**: Comprehensive error logging

**Logging**

* **Structured Logging**: JSON-formatted logs
* **Log Levels**: Error, Warning, Info, Debug
* **Console Output**: Development logging
* **File Logging**: Production log files

**Metrics**

* **Request Metrics**: API call counts and response times
* **Vector Search**: Pinecone query latency and success rates
* **Database Operations**: MongoDB query performance
* **User Analytics**: User behavior and conversion tracking

**Testing Metrics**

* **Load Testing Results**: Available in load-test-results.json
  + 600 virtual users created
  + 600 HTTP requests
  + 20 requests/second rate
  + Endpoints tested: Products, Search, Newsletter

**🚀 Deployment Configuration**

**Local Development**

* **Database**: MongoDB running on mongodb://localhost:27017
* **Backend**: Express server on http://localhost:5000
* **Frontend**: React dev server on http://localhost:3000
* **Hot Reload**: Nodemon for backend, React Fast Refresh for frontend
* **Proxy Configuration**: setupProxy.js for API requests

**Cloud Deployment**

**Frontend (Vercel)**

* **Platform**: Vercel
* **URL**: https://fusion-ecommerce-app.vercel.app
* **Build Command**: npm run build
* **Output Directory**: build
* **Environment Variables**: API URL configuration

**Backend (Vercel/Render)**

* **Primary**: Vercel Functions
* **Backup**: Render Web Service
* **Configuration**: backend/vercel.json
* **Database**: MongoDB Atlas (Cloud)
* **Environment Variables**:
  + MONGO\_URI: MongoDB connection string
  + JWT\_SECRET: JWT signing secret
  + PINECONE\_API\_KEY: Pinecone API key
  + PINECONE\_HOST: Pinecone index host
  + GOOGLE\_AI\_API\_KEY: Google AI API key

**Docker Deployment**

* **Dockerfile**: Multi-stage build for optimization
* **docker-compose.yml**: Service orchestration
* **Services**: Frontend, Backend, MongoDB
* **Networks**: Isolated container network
* **Volumes**: Persistent data storage

**Kubernetes Deployment**

* **Configuration Files**:
  + kubernetes/backend-deployment.yaml: Backend pods
  + kubernetes/backend-service.yaml: Backend service
  + kubernetes/frontend-deployment.yaml: Frontend pods
  + kubernetes/frontend-service.yaml: Frontend service
  + kubernetes/configmap.yaml: Configuration management
* **Scaling**: Horizontal Pod Autoscaler
* **Load Balancing**: Service-level load balancing

**📝 API Documentation**

**REST Endpoints**

**Authentication APIs**

* POST /api/auth/register - User registration (201 Created)
* POST /api/auth/login - User authentication (200 OK, 401 Unauthorized, 403 Forbidden)
* POST /api/auth/forgot-password - Initiate password reset
* POST /api/auth/reset-password - Complete password reset
* GET /api/auth/user - Get current user (Protected)

**Product APIs**

* GET /api/products - List all products (200 OK)
* GET /api/products/:id - Get product details (200 OK, 404 Not Found)
* GET /api/products/:id/similar - Get similar products (AI-powered)
* POST /api/products/recommendations - Get bulk recommendations
* GET /api/products/category/:category - Filter by category
* PUT /api/products/:id/rating - Update product rating

**Search APIs**

* GET /api/search?q=<query> - Search products (200 OK)

**Checkout APIs**

* POST /api/checkout/create-order - Create new order (200 OK, 400 Bad Request)

**Order APIs**

* POST /api/orders/track - Track order status (200 OK, 404 Not Found)
* GET /api/orders/:id - Get order details (Protected)

**Newsletter APIs**

* POST /api/newsletter/subscribe - Subscribe to newsletter (201 Created, 400 Bad Request)

**HTTP Status Codes**

| **Code** | **Name** | **Usage** |
| --- | --- | --- |
| 200 | OK | Successful request |
| 201 | Created | Resource created successfully |
| 400 | Bad Request | Invalid input or validation error |
| 401 | Unauthorized | Authentication failed |
| 403 | Forbidden | Access denied (locked account) |
| 404 | Not Found | Resource not found |
| 409 | Conflict | Resource already exists |
| 422 | Unprocessable Entity | Validation failed |
| 429 | Too Many Requests | Rate limit exceeded |
| 500 | Internal Server Error | Server failure |

**Swagger Documentation**

* **URL**: http://localhost:5000/api-docs
* **JSON**: http://localhost:5000/api-docs/swagger.json
* **Features**:
  + Interactive API testing
  + Request/response examples
  + Schema definitions
  + Error code documentation
  + Authentication examples

**🧪 Testing Strategy**

**Unit Testing**

* **Backend Tests**: backend/\_\_tests\_\_/
  + auth.spec.js: User registration, login, JWT validation
  + checkout.spec.js: Order creation, validation errors
  + products.spec.js: Product CRUD operations
  + search.spec.js: Product search queries
* **Frontend Tests**: src/tests/
  + Cart.test.js: Cart functionality
  + Checkout.test.js: Checkout process
  + Home.test.js: Homepage rendering
  + Login.test.js: Authentication flow
  + Register.test.js: User registration
  + Shop.test.js: Product listing

**Integration Testing**

* **API Endpoint Testing**: Using Supertest
* **Database Integration**: MongoDB connection and operations
* **Vector Database Integration**: Pinecone/Weaviate sync testing
* **Authentication Flow**: End-to-end auth testing

**End-to-End Testing**

* **User Workflows**: Complete user journeys
* **Cross-Browser Testing**: Chrome, Firefox, Safari
* **Responsive Testing**: Mobile, tablet, desktop
* **Performance Testing**: Load testing with Artillery

**Load Testing**

* **Tool**: Artillery.io
* **Results**: load-test-results.json
* **Metrics**:
  + Virtual Users: 600
  + HTTP Requests: 600
  + Request Rate: 20 req/sec
  + Endpoints: Products, Search, Newsletter

**Test Coverage**

* **Backend**: Jest with coverage reporting
* **Frontend**: React Testing Library with coverage
* **Target**: >80% code coverage
* **Reports**: Generated in /coverage directory

**📦 Project Structure**

MERN-Stack-Ecommerce-App/

├── backend/ # Node.js backend

│ ├── config/

│ │ └── db.js # MongoDB connection

│ ├── docs/

│ │ └── swagger.js # Swagger configuration

│ ├── models/

│ │ ├── product.js # Product schema with Pinecone hooks

│ │ ├── user.js # User schema

│ │ └── order.js # Order schema

│ ├── routes/

│ │ ├── auth.js # Authentication endpoints

│ │ ├── products.js # Product CRUD & recommendations

│ │ ├── search.js # Search functionality

│ │ ├── checkout.js # Order creation

│ │ ├── orders.js # Order tracking

│ │ └── newsletter.js # Newsletter subscription

│ ├── middleware/

│ │ └── auth.js # JWT verification

│ ├── scripts/

│ │ ├── build-faiss-index.js # FAISS index builder

│ │ ├── search-faiss-index.js # FAISS search

│ │ ├── sync-pinecone.js # Pinecone sync

│ │ ├── sync-weaviate-ids.js # Weaviate ID sync

│ │ ├── query-weaviate.js # Weaviate query

│ │ └── weaviate-upsert.js # Weaviate upsert

│ ├── seed/

│ │ └── productSeeds.js # Database seeding

│ ├── services/

│ │ └── pineconeSync.js # Pinecone sync helpers

│ ├── sync/

│ │ ├── syncPinecone.js # Main Pinecone sync

│ │ └── syncWeaviate.js # Weaviate sync

│ ├── \_\_tests\_\_/ # Backend tests

│ ├── faiss\_stores/ # FAISS index files

│ ├── pineconeClient.js # Pinecone SDK client

│ ├── weaviateClient.js # Weaviate SDK client

│ ├── index.js # Express server entry

│ ├── package.json # Backend dependencies

│ └── .env # Environment variables

│

├── src/ # React frontend

│ ├── components/

│ │ ├── CheckoutForm.jsx # Payment form

│ │ ├── Footer.jsx # Site footer

│ │ ├── NavigationBar.jsx # Top navigation

│ │ ├── ProductCard.jsx # Product tile

│ │ ├── ScrollToTop.jsx # Auto-scroll

│ │ └── SearchResults.jsx # Search display

│ ├── context/

│ │ └── NotificationProvider.jsx # Toast notifications

│ ├── pages/

│ │ ├── About.jsx # About page

│ │ ├── Cart.jsx # Shopping cart

│ │ ├── Checkout.jsx # Checkout page

│ │ ├── ForgotPassword.jsx # Password reset

│ │ ├── Home.jsx # Landing page

│ │ ├── Login.jsx # Login page

│ │ ├── NotFoundPage.jsx # 404 page

│ │ ├── OrderSuccess.jsx # Confirmation

│ │ ├── OrderTracking.jsx # Order tracking

│ │ ├── Privacy.jsx # Privacy policy

│ │ ├── ProductDetails.jsx # Product page

│ │ ├── Register.jsx # Registration

│ │ ├── ResetPassword.jsx # Reset password

│ │ ├── ShippingReturns.jsx # Shipping info

│ │ ├── Shop.jsx # All products

│ │ ├── Support.jsx # Support page

│ │ └── Terms.jsx # Terms of service

│ ├── services/

│ │ └── apiClient.js # Axios instance

│ ├── tests/ # Frontend tests

│ ├── utils/

│ │ └── products.js # Product helpers

│ ├── App.jsx # Root component

│ ├── App.css # Main styles

│ └── index.js # React entry

│

├── public/ # Static assets

│ ├── index.html # HTML template

│ ├── manifest.json # PWA manifest

│ └── favicon.ico # Favicon

│

├── docs/ # Documentation & screenshots

├── kubernetes/ # K8s configurations

├── terraform/ # Infrastructure as Code

├── vault/ # Secrets management

├── .github/workflows/ # CI/CD pipelines

│ └── ci.yml # GitHub Actions

│

├── docker-compose.yml # Docker orchestration

├── Dockerfile # Container definition

├── package.json # Frontend dependencies

├── craco.config.js # CRA overrides

├── jest.config.js # Jest configuration

├── setupProxy.js # Dev proxy

├── openapi.yaml # OpenAPI spec

├── AGENTS.md # AI agent instructions

├── API\_DOCUMENTATION.md # API docs

├── ARCHITECTURE.md # Architecture details

└── README.md # User documentation

**🔧 Configuration Management**

**Profile-Based Configuration**

**Development Environment** (backend/.env):

MONGO\_URI=mongodb://localhost:27017/Ecommerce-Products

JWT\_SECRET=development\_secret\_key

PORT=5000

PINECONE\_API\_KEY=pk-dev-xxx

GOOGLE\_AI\_API\_KEY=AIzaSy-dev-xxx

**Production Environment** (Cloud):

MONGO\_URI=mongodb+srv://user:pass@cluster.mongodb.net/production

JWT\_SECRET=secure\_production\_secret

PORT=8000

PINECONE\_API\_KEY=pk-prod-xxx

GOOGLE\_AI\_API\_KEY=AIzaSy-prod-xxx

**Environment Variables**

**Backend** (backend/.env):

* MONGO\_URI: MongoDB connection string
* JWT\_SECRET: JWT signing secret
* PORT: Server port (default: 5000)
* PINECONE\_API\_KEY: Pinecone authentication
* PINECONE\_HOST: Pinecone index URL
* PINECONE\_INDEX: Index name (e.g., ecommerce-products)
* PINECONE\_NAMESPACE: Namespace for vectors
* PINECONE\_PURGE\_ON\_SYNC: Clear vectors on sync (true/false)
* GOOGLE\_AI\_API\_KEY: Google AI for embeddings
* WEAVIATE\_HOST: Optional Weaviate instance URL
* WEAVIATE\_API\_KEY: Optional Weaviate authentication
* RECOMMENDATION\_PREFER\_WEAVIATE: Use Weaviate over Pinecone (true/false)

**Frontend** (Build-time):

* REACT\_APP\_API\_URL: Backend API base URL
* REACT\_APP\_ENVIRONMENT: Development/Production

**📋 Getting Started**

**Prerequisites**

* Node.js 18.x or higher
* MongoDB 6.x or higher
* npm 9.x or higher
* Git
* Pinecone account (for recommendations)
* Google AI API key (for embeddings)

**Local Development Setup**

1. **Clone the Repository**

git clone https://github.com/BITSSAP2025AugAPIBP3Sections/APIBP-20234YC-Team-09.git

cd APIBP-20234YC-Team-09

1. **Install Dependencies**

*# Install frontend dependencies*

npm install

*# Install backend dependencies*

cd backend

npm install

cd ..

1. **Configure Environment**

*# Create backend .env file*

cd backend

cp .env.example .env

*# Edit .env with your configuration*

cd ..

1. **Start MongoDB**

*# Windows*

net start MongoDB

*# Mac*

brew services start mongodb-community

*# Linux*

sudo systemctl start mongod

1. **Seed Database**

cd backend/seed

node productSeeds.js dev

cd ../..

1. **Sync Vector Databases**

cd backend

npm run sync-pinecone

*# Optional: npm run weaviate-upsert*

cd ..

1. **Start Development Servers**

*# Terminal 1: Backend*

cd backend

npm start

*# Terminal 2: Frontend*

npm start

1. **Access Application**

* Frontend: [http://localhost:3000](http://localhost:3000/)
* Backend API: [http://localhost:5000](http://localhost:5000/)
* Swagger Docs: <http://localhost:5000/api-docs>

**User Credentials (for testing)**

**Demo Account**:

* Email: [demo@fusion.com](mailto:demo@fusion.com)
* Password: demo123456

**Test Credit Card** (Checkout):

* Card Number: 4242 4242 4242 4242
* Expiry: Any future date (MM/YY)
* CVC: Any 3 digits

**🎓 DevOps Implementation**

**CI/CD Pipeline (GitHub Actions)**

**Workflow File**: .github/workflows/ci.yml

**Stages**:

1. **Build Stage**: Install dependencies and build application
2. **Test Stage**: Run unit and integration tests
3. **Lint Stage**: Code quality checks
4. **Security Scan**: Dependency vulnerability scanning
5. **Deploy Stage**: Automatic deployment to Vercel/Render

**Triggers**:

* Push to main branch
* Pull requests
* Manual workflow dispatch

**Docker Implementation**

**Multi-stage Dockerfile**:

* Stage 1: Build frontend
* Stage 2: Setup backend
* Stage 3: Production image

**Docker Compose Services**:

* Frontend (React)
* Backend (Node.js)
* MongoDB
* Nginx (reverse proxy)

**Kubernetes Orchestration**

**Resources**:

* Deployments for frontend and backend
* Services for load balancing
* ConfigMaps for configuration
* Persistent Volumes for data
* Horizontal Pod Autoscaler

**Namespace**: fusion-ecommerce

**Infrastructure as Code**

**Terraform** (terraform/main.tf):

* AWS/Cloud provider configuration
* Network setup
* Load balancer configuration
* Database provisioning

**Packer** (packer/fusion-electronics.pkr.hcl):

* Machine image creation
* Pre-configured application images

**📊 Monitoring & Logging (To Be Implemented)**

**ELK Stack Setup**

* **Elasticsearch**: Log storage and search
* **Logstash**: Log processing pipeline
* **Kibana**: Visualization dashboards

**Health Checks**

* Kubernetes liveness and readiness probes
* Application health endpoint
* Database connection monitoring

**Metrics**

* Request rate and latency
* Error rates
* Database query performance
* Vector search performance

**🔍 Code Quality & Static Analysis**

**SonarQube Integration (To Be Implemented)**

* Code smells detection
* Code coverage reports
* Security vulnerability scanning
* Technical debt tracking

**Code Quality Metrics**

* Maintainability rating
* Reliability rating
* Security rating
* Coverage percentage

**📚 Additional Documentation**

* **AGENTS.md**: AI coding agent instructions
* **API\_DOCUMENTATION.md**: Detailed API reference
* **ARCHITECTURE.md**: System architecture deep-dive
* **README.md**: User-facing documentation
* **SWAGGER\_ENHANCEMENTS.md**: Swagger improvements
* **USE\_CASE\_DIAGRAM.md**: Use case diagrams

**🤝 Team Information**

**Project**: APIBP-20234YC-Team-09  
**Institution**: BITS Pilani  
**Course**: API-Based Product Development (2023 4Y-C)  
**Team**: Team 22

**Team Members**:

* Suryansh Pandey ([suryansh.pandey@sap.com](mailto:suryansh.pandey@sap.com))
* [Add other team members]

**Repository**: <https://github.com/BITSSAP2025AugAPIBP3Sections/APIBP-20234YC-Team-09>

**📧 Contact & Support**

For issues, questions, or contributions:

* **GitHub Issues**: [Repository Issues](https://github.com/BITSSAP2025AugAPIBP3Sections/APIBP-20234YC-Team-09/issues)
* **Email**: [suryansh.pandey@sap.com](mailto:suryansh.pandey@sap.com)

**Last Updated**: November 30, 2025  
**Version**: 1.0.0  
**Status**: Production-Ready