PizzaWorld Dashboard

A comprehensive business intelligence dashboard system for PizzaWorld with Spring Boot backend, Angular frontend, and AI-powered assistant capabilities

Developed for the Programming Lab module in the Business Information Systems bachelor's program

Official Homepage:	https://www.pizzaworldplus.tech/
Dashboard Access:	https://dashboard.pizzaworldplus.tech/ (available via homepage or direct access)
Documentation:	https://www.pizzaworldplus.tech/documentation/ (available via homepage or direct access)
GitHub Repository:	https://github.com/luigids03/PizzaWorld
Contact:	pizzaworldplus@gmail.com

Demo Video

PizzaWorld Dashboard Demo - Watch our comprehensive demo video showcasing features and capabilities

Email Service Demo - Watch our email support system demonstration

Contents

1 Quick Start

1.1 System Requirements

- Java 17 or higher (OpenJDK or Oracle JDK)
- Node.js 18 or higher
- npm 9 or higher (included with Node.js)
- Modern web browser (Chrome, Firefox, Safari, or Edge)
- Operating System: Windows 10+, macOS 10.15+, or Linux

Note: PostgreSQL installation is not required - the application uses a pre-configured Supabase cloud database.

1.2 Security Implementation

This application demonstrates production-grade security practices:

- No hardcoded credentials in source code
- All sensitive data loaded from environment variables
- Application will not start without proper security configuration
- Start scripts handle secure environment setup automatically
- JWT-based authentication with role-based access control
- BCrypt password hashing for secure credential storage
- Google AI API key management for AI features

Important: Always use the provided start scripts. Direct execution will fail due to missing environment variables.

2 Installation and Startup

2.1 Windows Instructions

Listing 1: Windows Installation

```
# Clone the repository
git clone https://github.com/luigids03/PizzaWorld.git
cd PizzaWorld

# Start both Backend and Frontend
./start.bat
```

2.2 macOS/Linux Instructions

Listing 2: macOS/Linux Installation

```
# Clone the repository
git clone https://github.com/luigids03/PizzaWorld.git
cd PizzaWorld

# Make script executable (first time only)
chmod +x start.sh

# Start both Backend and Frontend
./start.sh
```

The script automatically:

- Sets all required environment variables securely
- Configures JVM memory optimization (512MB-2GB)
- Starts the Spring Boot backend (port 8080)
- Starts the Angular frontend (port 4200)
- Opens http://localhost:4200 in your default browser

3 Features Overview

3.1 Backend (Spring Boot)

- **RESTful API** with 90+ endpoints covering all business operations
- **AI Assistant Integration** with Google Gemma AI and real-time streaming
- **JWT Authentication** with role-based access control (RBAC)
- **Spring Security** with custom authentication filter
- **Supabase PostgreSQL Integration** with optimized native queries
- **Materialized Views** for high-performance analytics (15+ specialized views)
- **Advanced Store Analytics** with individual store performance tracking
- **Customer Analytics** including lifetime value and retention analysis
- **Product Analytics** with trend analysis and comparison capabilities
- **Revenue Analytics** with multi-dimensional filtering and time series
- **Email Support System** with SMTP integration for customer support
- **Knowledge Base** with document retrieval and contextual responses
- **Comprehensive CSV Export** for all data views with role-based filtering
- **Dynamic Filtering & Pagination** for large datasets
- **Global Exception Handling** with meaningful error messages

3.2 Frontend (Angular 19)

- **Responsive Design** with Tailwind CSS
- **Interactive Dashboards** with ApexCharts visualizations
- **AI Chatbot Interface** with real-time streaming responses
- **Real-time Updates** via efficient HTTP polling
- **Role-based UI** with dynamic navigation
- **Progressive Web App** capabilities
- **TypeScript** for type safety
- **Component Architecture** with lazy loading
- **State Management** with RxJS

3.3 AI & Intelligence Features

- **Google Gemma AI Integration** for intelligent responses
- **Real-time Chat Streaming** using Server-Sent Events
- **Business Context Integration** with live data
- **Knowledge Base Retrieval** for contextual responses
- **Natural Language Analytics** for business queries
- **AI-generated Insights** based on business data
- **Role-based AI Responses** tailored to user permissions
- **Intelligent Fallbacks** when AI is unavailable

3.4 Advanced Store Analytics

- **Individual Store Performance** tracking with comprehensive KPIs
- **Store Revenue Trends** with customizable time periods and comparisons
- **Hourly Performance Analysis** for operational optimization
- **Category Performance Tracking** by store location
- **Daily Operations Monitoring** with efficiency metrics
- **Customer Insights** per store including acquisition and retention
- **Product Performance Analysis** by individual store
- **Store Efficiency Metrics** including capacity utilization
- **Comparative Analytics** between stores, states, and national averages
- **Custom Date Range Analysis** with flexible time period selection

3.5 Customer Analytics & Intelligence

- **Customer Lifetime Value (CLV)** analysis with segmentation
- **Customer Retention Analysis** with cohort tracking
- **Customer Acquisition Metrics** by location and time period
- **Store Capacity Analysis** including peak hours identification
- **Customer Distance Metrics** for delivery optimization
- **Retention Rate Tracking** across different customer segments
- **Purchase Pattern Analysis** for targeted marketing

3.6 Advanced Product Analytics

- **Product Performance Tracking** with trend analysis
- **Product KPI Analysis** including revenue, orders, and growth metrics
- **Product Comparison Tools** for competitive analysis
- **Revenue by Category** breakdown with filtering capabilities
- **Product Trend Analysis** with customizable time intervals
- **Launch Performance Tracking** for new products
- **Category Performance Analytics** across all product lines
- **Product Mix Analysis** by store and region

3.7 Data Export & Reporting

- **Comprehensive CSV Export** for all analytics views and data tables
- **Role-based Export Filtering** ensuring users only export permitted data
- **Dashboard KPI Export** with summary metrics
- **Store Performance Export** including all analytics and comparisons
- **Product Performance Export** with trend analysis and comparisons
- **Customer Analytics Export** including CLV and retention data
- **Revenue Analytics Export** with time series and comparative data
- **Orders Export** with advanced filtering and search capabilities
- **Store Capacity Export** with utilization and efficiency metrics
- **Custom Date Range Export** for any time period analysis
- **Automated Export Processing** with optimized data formatting

3.8 Email Support System

- **Customer Support Integration** with automated email processing
- **Background Email Processing** for improved response times
- **Support Ticket Management** with email notifications
- **SMTP Integration** with Gmail for reliable delivery
- **Error Handling & Logging** for support request tracking

4 User Roles & Permissions

Role	D	O	P	S	A	E	AI	Ad
HQ_ADMIN	✓	✓	✓	✓	✓	✓	✓	✓
STATE_MANAGER	✓	✓	✓	✓	✓	✓	✓	×
STORE_MANAGER	✓	✓	~	✓	✓	✓	✓	×

Legend:

- **D** = Dashboard Access
- **O** = Orders Management
- **P** = Products Management
- **S** = Stores Management
- **A** = Analytics Access
- **E** = Export Functionality
- **AI** = AI Assistant Access
- **Ad** = Admin Functions

Permission Levels:

- ✓ = Full Access
- ~ = Limited Access (View Only)
- × = No Access

Role Details:

- **HQ_ADMIN**: Complete system access including user management and full CRUD operations
- **STATE_MANAGER**: State-level data access with full view/edit permissions for products
- **STORE_MANAGER**: Store-level data access with view-only access for products

5 API Endpoints

5.1 Authentication

- `POST /api/login` - User authentication
- `GET /api/me` - Current user information
- `POST /api/logout` - User logout
- `POST /api/create-test-user` - Create test user (development)

5.2 Dashboard & Analytics

- `GET /api/v2/dashboard/kpis` - Main KPI metrics
- `GET /api/v2/dashboard/kpis/export` - Export dashboard KPIs to CSV
- `GET /api/v2/orders/recent` - Recent order list
- `GET /api/v2/analytics/revenue/by-year` - Annual revenue analytics
- `GET /api/v2/analytics/revenue/by-month` - Monthly revenue analytics
- `GET /api/v2/analytics/revenue/by-week` - Weekly revenue analytics
- `GET /api/v2/analytics/revenue/by-store` - Revenue by store analytics
- `GET /api/v2/analytics/store-performance` - Store performance metrics
- `GET /api/v2/analytics/hourly-performance` - Hourly performance analytics
- `GET /api/v2/analytics/hourly-performance/export` - Export hourly performance data
- `GET /api/v2/analytics/product-performance` - Product performance analytics
- `GET /api/v2/analytics/product-performance/export` - Export product performance data
- `GET /api/v2/analytics/category-performance` - Category performance analytics
- `GET /api/v2/analytics/daily-trends` - Daily revenue trends
- `GET /api/v2/analytics/monthly-trends` - Monthly revenue trends
- `GET /api/v2/analytics/store-comparison` - Store performance comparison
- `GET /api/v2/analytics/state-comparison` - State revenue trends
- `GET /api/v2/analytics/state-performance` - State performance analytics
- `GET /api/v2/analytics/monthly-revenue-trends` - Monthly revenue trends by store

5.3 Store Analytics

- `GET /api/v2/stores` - Store directory with role-based filtering
- `GET /api/v2/stores/export` - Export stores to CSV
- `GET /api/v2/stores/performance` - Store performance analytics
- `GET /api/v2/stores/performance/export` - Export store performance to CSV
- `GET /api/v2/stores/hourly-performance` - Aggregated hourly performance across stores
- `GET /api/v2/stores/customer-acquisition` - Store customer acquisition metrics
- `GET /api/v2/stores/product-mix` - Product mix analysis by store
- `GET /api/v2/stores/weekly-trends` - Weekly performance trends
- `GET /api/v2/stores/daily-operations` - Daily operations overview
- `GET /api/v2/stores/efficiency-metrics` - Store efficiency metrics

5.4 Individual Store Analytics

- `GET /api/v2/stores/{storeId}/analytics/overview` - Individual store overview
- `GET /api/v2/stores/{storeId}/analytics/revenue-trends` - Store revenue trends
- `GET /api/v2/stores/{storeId}/analytics/hourly-performance` - Store hourly performance
- `GET /api/v2/stores/{storeId}/analytics/category-performance` - Store category performance
- `GET /api/v2/stores/{storeId}/analytics/daily-operations` - Store daily operations
- `GET /api/v2/stores/{storeId}/analytics/customer-insights` - Store customer insights
- `GET /api/v2/stores/{storeId}/analytics/product-performance` - Store product performance
- `GET /api/v2/stores/{storeId}/analytics/recent-orders` - Store recent orders
- `GET /api/v2/stores/{storeId}/analytics/efficiency-metrics` - Store efficiency metrics
- `GET /api/v2/stores/{storeId}/analytics/contextual-overview` - Enhanced contextual overview
- `GET /api/v2/stores/{storeId}/analytics/enhanced-revenue-trends` - Enhanced revenue trends
- `GET /api/v2/stores/{storeId}/analytics/enhanced-performance` - Enhanced performance metrics
- `GET /api/v2/stores/{storeId}/analytics/custom-range` - Custom date range analytics
- `POST /api/v2/stores/{storeId}/analytics/compare` - Compare analytics periods

5.5 Advanced Product Analytics

- `GET /api/v2/products` - Product catalog with search and filtering
- `GET /api/v2/products/export` - Export products to CSV
- `GET /api/v2/products/top` - Top performing products
- `GET /api/v2/products/top/export` - Export top products to CSV
- `GET /api/v2/products/performance` - Product performance analytics
- `GET /api/v2/products/performance/export` - Export product performance to CSV
- `GET /api/v2/products/kpis` - Product KPIs with filtering
- `GET /api/v2/products/revenue-by-category` - Revenue breakdown by category
- `GET /api/v2/products/details/{sku}` - Product details by SKU

5.6 Individual Product Analytics

- `GET /api/v2/products/kpi` - Individual product KPI analysis
- `GET /api/v2/products/trend` - Product trend analysis
- `GET /api/v2/products/trend/export` - Export product trend data to CSV
- `GET /api/v2/products/compare` - Product comparison analysis
- `GET /api/v2/products/revenue-trend` - Product revenue trend analysis

5.7 Product Chart Analytics

- `GET /api/v2/products/overview-chart` - Products overview chart data
- `GET /api/v2/products/overview-chart/custom-range` - Custom range overview chart
- `POST /api/v2/products/overview-chart/compare` - Compare overview chart periods
- `GET /api/v2/products/analytics/custom-range` - Custom range product analytics
- `GET /api/v2/products/analytics/custom-range/export` - Export custom range analytics
- `POST /api/v2/products/analytics/compare` - Compare product analytics periods
- `POST /api/v2/products/analytics/compare/export` - Export analytics comparison

5.8 Customer Analytics

- `GET /api/v2/analytics/customer-lifetime-value` - Customer lifetime value analysis
- `GET /api/v2/analytics/customer-lifetime-value/summary` - CLV summary metrics
- `GET /api/v2/analytics/customer-lifetime-value/export` - Export CLV data to CSV
- `GET /api/v2/analytics/customer-retention` - Customer retention analysis
- `GET /api/v2/analytics/customer-retention/export` - Export retention data to CSV
- `GET /api/v2/analytics/customer-acquisition` - Customer acquisition analytics

- `GET /api/v2/analytics/store-capacity` - Store capacity analysis (legacy)
- `GET /api/v2/analytics/store-capacity/summary` - Store capacity summary (legacy)
- `GET /api/v2/analytics/store-capacity/peak-hours` - Peak hours analysis (legacy)
- `GET /api/v2/analytics/store-capacity/export` - Export capacity data (legacy)

5.9 Advanced Store Capacity Analytics (V3)

- `GET /api/v2/analytics/store-capacity-v3/summary` - Enhanced store capacity summary
- `GET /api/v2/analytics/store-capacity-v3/metrics` - Store capacity metrics
- `GET /api/v2/analytics/store-capacity-v3/peak-hours` - Advanced peak hours analysis
- `GET /api/v2/analytics/store-capacity-v3/customer-distance` - Customer distance metrics
- `GET /api/v2/analytics/store-capacity-v3/delivery-metrics` - Delivery performance metrics
- `GET /api/v2/analytics/store-capacity-v3/utilization-chart` - Store utilization chart data
- `GET /api/v2/analytics/store-capacity-v3/export` - Export enhanced capacity data

5.10 Orders Management

- `GET /api/v2/orders` - Orders with advanced filtering and pagination
- `GET /api/v2/orders/export` - Export filtered orders to CSV
- `GET /api/v2/orders/available-states` - Available states for filtering
- `GET /api/v2/orders/kpis` - Orders KPIs with filtering

5.11 Global KPI Analytics

- `GET /api/v2/kpis/global-store` - Global store KPIs from materialized views
- `GET /api/v2/kpis/global-store/export` - Export global store KPIs to CSV

5.12 Revenue Analytics & Charts

- `GET /api/v2/store-revenue-chart` - Store revenue chart data
- `GET /api/v2/store-revenue-chart/export` - Export revenue chart to CSV
- `GET /api/v2/store-revenue-chart/years` - Available years for charts
- `GET /api/v2/store-revenue-chart/months` - Available months for charts
- `GET /api/v2/chart/store-revenue` - Dynamic store revenue charts
- `GET /api/v2/chart/store-revenue/date-range` - Revenue by custom date range
- `GET /api/v2/chart/store-revenue/export` - Export chart data to CSV

5.13 Chart Time Period Utilities

- `GET /api/v2/chart/time-periods/years` - Available years for time filtering
- `GET /api/v2/chart/time-periods/months` - Available months for year
- `GET /api/v2/chart/time-periods/quarters` - Available quarters for year

5.14 AI Assistant

- `POST /api/ai/chat` - AI chat interaction
- `POST /api/ai/chat/stream` - Streaming AI responses
- `GET /api/ai/chat/history/{sessionId}` - Get chat history
- `POST /api/ai/analyze` - Natural language query analysis
- `GET /api/ai/insights` - AI-generated business insights
- `GET /api/ai/health` - AI service health check
- `GET /api/ai/config` - Public AI configuration
- `GET /api/ai/status` - AI system status
- `POST /api/ai/test` - Test Google AI connection

5.15 Email Support

- `POST /api/send-support-email` - Send support email

5.16 System Health

- `GET /api/v2/health` - API health check
- `GET /api/v2/profile` - User profile information

6 Technology Stack

6.1 Backend Technologies

- **Spring Boot 3.4.6** - Application framework
- **Spring Security 6** - Authentication & authorization
- **Spring Data JPA** - ORM layer
- **Spring Mail** - Email integration
- **Spring WebFlux** - Reactive programming for AI integration
- **Supabase PostgreSQL** - Cloud database
- **HikariCP** - Connection pooling
- **JWT** - Authentication tokens
- **Maven** - Build automation
- **Java 17** - Runtime environment

6.2 Frontend Technologies

- **Angular 19.1.0** - SPA framework
- **TypeScript 5.7.2** - Type-safe JavaScript
- **RxJS 7.8.0** - Reactive programming
- **Tailwind CSS 3.4.17** - Utility-first CSS
- **PrimeNG 19.1.3** - UI component library
- **ApexCharts 3.41.0** - Data visualization
- **Angular Material 19.2.18** - Material Design components
- **Chart.js 4.4.9** - Additional charting library
- **D3 7.9.0** - Data visualization utilities

6.3 AI & Integration

- **Google Gemma AI** - Language model integration
- **Server-Sent Events** - Real-time streaming
- **WebClient** - HTTP client for AI API calls

7 Configuration

7.1 Environment Variables

The application requires the following environment variables for security:

Variable	Description	Required
DB_URL	Supabase PostgreSQL connection URL	Yes
DB_USERNAME	Database username	Yes
DB_PASSWORD	Database password	Yes
JWT_SECRET	Secret key for JWT signing	Yes
GMAIL_APP_PASSWORD	Gmail app password for email	Yes
GOOGLE_AI_API_KEY	Google AI API key	Yes
GOOGLE_AI_MODEL	Google AI model name	No

Security Note: All sensitive credentials are managed through environment variables and are never hardcoded in the source code.

7.2 Database Configuration

The application uses a **Supabase PostgreSQL** cloud database with:

- SSL/TLS encryption
- Connection pooling (30 max connections)
- Optimized queries with materialized views
- Role-based data access control

8 Deployment Options

8.1 Local Development

Listing 3: Local Development Commands

```
# Windows
./start.bat

# macOS/Linux
./start.sh
```

8.2 Docker Deployment

Listing 4: Docker Deployment

```
# Build and run with Docker
docker build -t pizzaworld-app .
docker run -p 8080:8080 pizzaworld-app
```

8.3 Cloud Deployment

The application includes configuration for Render.com cloud deployment with automatic environment setup.

9 Development

9.1 Backend Development

Listing 5: Backend Development Commands

```
# Run in development mode
./mvnw spring-boot:run

# Run tests
./mvnw test

# Build JAR
./mvnw clean package
```

9.2 Frontend Development

Listing 6: Frontend Development Commands

```
# Development server with hot reload
ng serve

# Production build
ng build --configuration production

# Run unit tests
ng test
```

10 Troubleshooting

10.1 Common Issues

10.1.1 Backend won't start

- Ensure using start scripts (`start.bat` / `start.sh`)
- Verify Java 17+ installed: `java -version`
- Check environment variables are set

10.1.2 Frontend compilation errors

- Clear node_modules: `rm -rf node_modules package-lock.json`
- Reinstall dependencies: `npm install`
- Check Node.js version: `node -v` (should be 18+)

10.1.3 AI features not working

- Verify Google AI API key configuration
- Check AI service status: `GET /api/ai/status`
- Test AI connectivity: `POST /api/ai/test`

11 Demo Data

The application includes comprehensive demo data:

- **4 US States:** Arizona, California, Nevada, Utah
- **52 Stores:** Distributed across states
- **100,000+ Orders:** 3 years of historical data (2021-2023)
- **25+ Products:** Various pizza types and sizes
- **Performance Metrics:** Pre-calculated analytics

12 Academic Context

This application was developed as part of the Programming Lab module in the Business Information Systems bachelor's program. It demonstrates:

12.1 Software Engineering

- Clean architecture with separation of concerns
- Design patterns (Repository, DTO, Factory, Strategy)
- SOLID principles application
- Comprehensive error handling and logging

12.2 Full-Stack Development

- RESTful API design and implementation
- Single Page Application architecture
- Responsive web design principles
- State management with reactive programming

12.3 Advanced Technologies

- AI integration with Google Gemma
- Real-time features with WebSocket and SSE
- Cloud services and external API integration
- Containerization with Docker

12.4 Security & Performance

- JWT authentication with role-based access
- Environment-based secure configuration
- Database query optimization with materialized views
- Caching strategies and performance monitoring