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/
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

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 comprehensive endpoint coverage
- **AI Assistant Integration** with Google Gemma AI
- **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
- **Email System** with SMTP integration and support tickets
- **Knowledge Base** with document retrieval and contextual responses
- **CSV Export** for all data tables with role-based filtering
- **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

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

5.2 Dashboard & Analytics

- `GET /api/dashboard-kpis` - Main KPI metrics
- `GET /api/recent-orders` - Recent order list
- `GET /api/kpi/revenue-trend` - Revenue analytics
- `GET /api/analytics/customer-lifetime-value` - CLV analysis
- `GET /api/analytics/customer-retention` - Retention metrics

5.3 AI Assistant

- `POST /api/ai/chat` - AI chat interaction
- `POST /api/ai/chat/stream` - Streaming AI responses
- `POST /api/ai/analyze` - Natural language query analysis
- `GET /api/ai/insights` - AI-generated business insights
- `GET /api/ai/status` - AI system status

5.4 Business Operations

- `GET /api/orders` - Order management with filtering
- `GET /api/products` - Product catalog
- `GET /api/stores` - Store directory
- `GET /api/customers` - Customer data

5.5 Data Export

- `GET /api/orders/export` - Orders CSV export
- `GET /api/products/export` - Products CSV export
- `GET /api/stores/export` - Stores CSV export

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** - SPA framework
- **TypeScript 5.7** - Type-safe JavaScript
- **RxJS 7.8** - Reactive programming
- **Tailwind CSS 3.4** - Utility-first CSS
- **PrimeNG 19** - UI component library
- **ApexCharts 3.41** - Data visualization
- **Angular Material 19** - Material Design components

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