Food Ordering System - Localhost Deployment Report

System Overview

Food Ordering System is a microservices application consisting of 4 main components:

- **User Service** (Django/Python): User management and JWT authentication
- **Menu Service** (Node.js/Express): Food menu management
- **Order Service** (Node.js/Express): Order management
- **Frontend** (React/TypeScript/Vite): User interface

System Requirements

Required Software

- **Node.js** (v18+)
- **Python** (v3.11+)
- **MongoDB** (v6+)
- **PostgreSQL** (v15+)
- **Redis** (v7+)
- **Git**

Software Installation

Windows

```
# Node.js
# Download from https://nodejs.org/

# Python
# Download from https://www.python.org/downloads/

# MongoDB
# Download from https://www.mongodb.com/try/download/community

# PostgreSQL
# Download from https://www.postgresql.org/download/windows/

# Redis
# Download from https://github.com/microsoftarchive/redis/releases
```

Linux (Ubuntu/Debian)

```
# Node.js
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs

# Python
sudo apt-get update
sudo apt-get install python3.11 python3-pip

# MongoDB
wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/6.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list
sudo apt-get update
sudo apt-get install -y mongodb-org

# PostgreSQL
sudo apt-get install postgresql postgresql-contrib

# Redis
sudo apt-get install redis-server
```

Installation and Configuration

1. Clone repository

```
git clone <repository-url>
cd Food-Ordering-System
```

2. Database Configuration

MongoDB

```
# Start MongoDB
sudo systemctl start mongod
sudo systemctl enable mongod

# Create databases
mongosh
use food_ordering_menus
use food_ordering_orders
```

PostgreSQL

```
# Start PostgreSQL
sudo systemctl start postgresql
sudo systemctl enable postgresql

# Create database and user
sudo -u postgres psql
CREATE DATABASE food_ordering_users;
CREATE USER postgres WITH PASSWORD 'password';
GRANT ALL PRIVILEGES ON DATABASE food_ordering_users TO postgres;
\q
```

Redis

```
# Start Redis
sudo systemctl start redis-server
sudo systemctl enable redis-server
```

3. Install User Service (Django)

```
cd backend/user-service

# Create virtual environment
python -m venv venv
source venv/bin/activate  # Linux/Mac
# or
venv\Scripts\activate     # Windows

# Install dependencies
pip install -r requirements.txt

# Create .env file
cat > .env << EOF
DEBUG=True
SECRET_KEY=django-insecure-development-key-change-in-production
DB_NAME=food_ordering_users
DB_USER=postgres
DB_PASSWORD=password
DB_HOST=localhost
DB_PORT=5432
REDIS_URL=redis://localhost:6379/0
ALLOWED_HOSTS=localhost,127.0.0.1
CORS_ALLOWED_ORIGINS=http://localhost:3000,http://127.0.0.1:3000
EOF

# Run migrations
python manage.py makemigrations
python manage.py migrate

# Create superuser (optional)
python manage.py createsuperuser

# Start service
python manage.py runserver 0.0.0.0:8000
```

4. Install Menu Service (Node.js)

```
cd backend/menu-service

# Install dependencies
npm install

# Create .env file
cat > .env << EOF
NODE_ENV=development
PORT=3001
MONGODB_URI=mongodb://localhost:27017/food_ordering_menus
USER_SERVICE_URL=http://localhost:8000
ORDER_SERVICE_URL=http://localhost:3002
EOF

# Start service
npm run dev
```

5. Install Order Service (Node.js)

```
cd backend/order-service

# Install dependencies
npm install

# Create .env file
cat > .env << EOF
NODE_ENV=development
PORT=3002
MONGODB_URI=mongodb://localhost:27017/food_ordering_orders
USER_SERVICE_URL=http://localhost:8000
MENU_SERVICE_URL=http://localhost:3001
EOF

# Start service
npm run dev
```

6. Install Frontend (React)

```
cd frontend/restaurant

# Install dependencies
npm install

# Create .env file
cat > .env << EOF
VITE_API_URL=http://localhost:8000/api
VITE_MENU_SERVICE_URL=http://localhost:3001
VITE_ORDER_SERVICE_URL=http://localhost:3002
VITE_DEV_MODE=true
EOF

# Start development server
npm run dev
```

Automation Scripts

Windows (run-all-services.bat)

```
# Start all services
run-all-services.bat start

# Check status
run-all-services.bat status

# View logs
run-all-services.bat logs

# Stop all services
run-all-services.bat stop
```

Linux/Mac (run-all-services.sh)

```
# Make executable
chmod +x run-all-services.sh

# Start all services
./run-all-services.sh start

# Check status
./run-all-services.sh status

# View logs
./run-all-services.sh logs

# Stop all services
./run-all-services.sh stop
```

Health Checks

Health Check Endpoints

```
# User Service
curl http://localhost:8000/api/health/

# Menu Service
curl http://localhost:3001/health

# Order Service
curl http://localhost:3002/health

# Frontend
curl http://localhost:3000
```

Access URLs

- **Frontend**: http://localhost:3000
- **User Service API**: http://localhost:8000/api
- **Menu Service API**: http://localhost:3001/api
- **Order Service API**: http://localhost:3002/api
- **Django Admin**: http://localhost:8000/admin

Environment Configuration

Important Environment Variables

User Service (.env)

```
DEBUG=True
SECRET_KEY=your-secret-key
DB_NAME=food_ordering_users
DB_USER=postgres
DB_PASSWORD=password
DB_HOST=localhost
DB_PORT=5432
REDIS_URL=redis://localhost:6379/0
CORS_ALLOWED_ORIGINS=http://localhost:3000
```

Menu Service (.env)

```
NODE_ENV=development
PORT=3001
MONGODB_URI=mongodb://localhost:27017/food_ordering_menus
USER_SERVICE_URL=http://localhost:8000
ORDER_SERVICE_URL=http://localhost:3002
```

Order Service (.env)

```
NODE_ENV=development
PORT=3002
MONGODB_URI=mongodb://localhost:27017/food_ordering_orders
USER_SERVICE_URL=http://localhost:8000
```

```
MENU_SERVICE_URL=http://localhost:3001
```

Frontend (.env)

```
VITE_API_URL=http://localhost:8000/api
VITE_MENU_SERVICE_URL=http://localhost:3001
VITE_ORDER_SERVICE_URL=http://localhost:3002
VITE_DEV_MODE=true
```

Troubleshooting

Common Issues

1. Port already in use

```
# Check which ports are in use
netstat -tulpn | grep :3000
netstat -tulpn | grep :3001
netstat -tulpn | grep :3002
netstat -tulpn | grep :8000

# Kill process using port
sudo kill -9 <PID>
```

2. Database connection failed

```
# Check MongoDB
sudo systemctl status mongod

# Check PostgreSQL
sudo systemctl status postgresql

# Check Redis
sudo systemctl status redis-server
```

3. CORS errors

- Ensure CORS_ALLOWED_ORIGINS is configured correctly
- Check frontend URL in backend configuration

4. JWT token errors

- Check SECRET_KEY in User Service
- Ensure system time is accurate

Logs and Debugging

View service logs

```
# User Service
cd backend/user-service
tail -f logs/django.log

# Menu Service
cd backend/menu-service
npm run dev

# Order Service
cd backend/order-service
npm run dev

# Frontend
cd frontend/restaurant
npm run dev
```

Security

Production Security Configuration

1. Change SECRET_KEY
2. Disable DEBUG mode
3. Configure HTTPS
4. Use strong passwords
5. Configure firewall
6. Regular security updates

Backup and Recovery

```
# Backup MongoDB
mongodump --db food_ordering_menus --out backup/
mongodump --db food_ordering_orders --out backup/

# Backup PostgreSQL
pg_dump food_ordering_users > backup/users_backup.sql

# Restore
mongorestore --db food_ordering_menus backup/food_ordering_menus/
psql food_ordering_users < backup/users_backup.sql
```

Monitoring and Performance

Monitoring Tools

- **Application**: Built-in health checks
- **Database**: MongoDB Compass, pgAdmin
- **System**: htop, iotop, netstat

Performance Optimization

1. Database indexing
2. Redis caching
3. Load balancing
4. CDN for static files
5. Compression

Conclusion

The Food Ordering System has been successfully deployed on localhost with all microservices. All services have health checks and logging for easy monitoring and debugging.

To run the system, simply:

1. Install dependencies
2. Configure databases
3. Run script `run-all-services.bat` (Windows) or `run-all-services.sh` (Linux/Mac)

The system will automatically start all services in the appropriate order and check health status.