Auctionality

Auctionality is a full-stack auction platform composed of a Spring Boot backend, a
 RabbitMQ infrastructure layer, and a React/Vite frontend. This guide explains how
 every piece locallyâ from configuring secrets and databases to running the monit
you can start developing or demoing the product quickly.

Repository Structure

- ‘backend/‘ â Spring Boot 4.0 project \(Java 25, Maven Wrapper\) exposing REST,
OAuth2, email, and payment integrations.
- ‘frontend/‘ â React 19 + TypeScript + Vite client, Tailwind + MUI UI, Redux to
- ‘monitoring/‘ & ‘docker-compose.monitoring.yml‘ â Prometheus + Grafana stack w
Boot Actuator metrics.
- ‘docs/‘ â Product proposal, diagrams, and integration notes.
- ‘logs/‘ â Default target for backend logback appenders \(rotated daily\).
- ‘GOOGLE_OAUTH2_SETUP.md‘, ‘MONITORING_SETUP.md‘, ‘MONITORING_AND_LOGGING.md‘ â
guides referenced throughout this README.

Prerequisites

Tool / Service	Required Version	Notes
Java Development Kit	25 \(Temurin or Oracle\)	Backend uses preview features;
accept ‘--enable-preview‘.
Maven Wrapper \(‘./mvnw‘\)	Bundled	No global Maven installation needed.
Node.js / npm	Node 20+ / npm 10+	For Vite dev server and builds.
PostgreSQL	14+	Stores all domain data; auto-seeding SQL runs on app startup.
RabbitMQ	3.13+ \(with management plugin\)	Powers real-time bid notifications
queues.
Docker + Docker Compose	Latest	Optional but recommended for Postgres/RabbitM
and the monitoring stack.
SendGrid account	Optional	Needed to send transactional emails.
Google Cloud project	Optional	Required for Google OAuth2 login.
VNPAY sandbox credentials	Optional	Needed only when testing VNPAY checkout.

> **Tip:** macOS developers can install the basics with Homebrew \(‘brew install t

Quick Start \(Local Development\)

1. **Clone and enter the repo**
   ‘‘‘bash
   git clone <your-fork-url>
   cd Auctionality

```
   ‘‘‘
2. **Start local infrastructure** \(use Docker or your preferred services\):
   ‘‘‘bash
   docker run --name auctionality-postgres -e POSTGRES_DB=auctionality \\
     -e POSTGRES_USER=auctionality -e POSTGRES_PASSWORD=auctionality \\
     -p 5432:5432 -d postgres:15

   docker run --name auctionality-rabbit -p 5672:5672 -p 15672:15672 \\
     -d rabbitmq:3.13-management
   ‘‘‘
3. **Configure backend secrets** â   populate ‘backend/src/main/resources/secrets.
\(see [Backend configuration]\(#backend-spring-boot\)\).
4. **Run the backend**
   ‘‘‘bash
   cd backend
   ./mvnw spring-boot:run -Dspring-boot.run.jvmArguments=--enable-preview
   ‘‘‘
   API root: ‘http://localhost:8081/api‘, Swagger: ‘http://localhost:8081/swagger-
5. **Configure the frontend env** â   create ‘frontend/.env.local‘ with API/WS URL
6. **Run the frontend**
   ‘‘‘bash
   cd frontend
   npm install
   npm run dev
   ‘‘‘
   App served at ‘http://localhost:5173‘.
7. **\(Optional\) Start monitoring**
   ‘‘‘bash
   docker-compose -f docker-compose.monitoring.yml up -d
   ‘‘‘
   Prometheus: ‘http://localhost:9090‘, Grafana: ‘http://localhost:3000‘.

## Backend \(Spring Boot\)

### Configuration & Secrets

The backend loads configuration from ‘application.yml‘ and overlays secrets via ei
environment variables or ‘backend/src/main/resources/secrets.properties‘ \(ignored
Create that file if it does not exist:

‘‘‘properties
# Database
DB_HOST=localhost
DB_PORT=5432
```

```
DB_NAME=auctionality
DB_USERNAME=auctionality
DB_PASSWORD=super-secret

# Security
JWT_SECRET=replace-with-32-char-minimum-value
FRONTEND_BASE_URL=http://localhost:5173

# Integrations
GOOGLE_CLIENT_ID=...
GOOGLE_CLIENT_SECRET=...
SENDGRID_API_KEY=...
SENDGRID_FROM_EMAIL=no-reply@auctionality.local
SENDGRID_FROM_NAME=Auctionality
RECAPTCHA_SITE_KEY=...
RECAPTCHA_SECRET_KEY=...
VNPAY_TMN_CODE=...
VNPAY_SECRET_KEY=...
‘‘‘

Core required variables:

| Key | Description | Default |
| --- | --- | --- |
| ‘DB_HOST‘, ‘DB_PORT‘, ‘DB_NAME‘, ‘DB_USERNAME‘, ‘DB_PASSWORD‘ | PostgreSQL conne
 | None |
| ‘JWT_SECRET‘ | 32+ char signing key for access/refresh tokens | Placeholder in
‘application.yml‘ \(replace immediately\) |
| ‘FRONTEND_BASE_URL‘ | Used when generating email links and OAuth redirects |
‘http://localhost:5173‘ |
| ‘RABBITMQ_HOST‘, ‘RABBITMQ_PORT‘, ‘RABBITMQ_USERNAME‘, ‘RABBITMQ_PASSWORD‘ | Rab
connection \(defaults to ‘localhost:5672‘, ‘guest/guest‘\) | Provided in ‘applicat

Optional integrations:

| Key | Purpose |
| --- | --- |
| ‘SENDGRID_API_KEY‘, ‘SENDGRID_FROM_EMAIL‘, ‘SENDGRID_FROM_NAME‘ | Email verifica
transactional notifications. |
| ‘GOOGLE_CLIENT_ID‘, ‘GOOGLE_CLIENT_SECRET‘ | Enables Google OAuth2 login \(see )
| ‘RECAPTCHA_SITE_KEY‘, ‘RECAPTCHA_SECRET_KEY‘, ‘RECAPTCHA_ENABLED‘ | reCAPTCHA gu
signup. |
| ‘VNPAY_URL‘, ‘VNPAY_TMN_CODE‘, ‘VNPAY_SECRET_KEY‘, ‘VNPAY_USD_TO_VND_RATE‘ | Pay
integration. |
```

| ‘SENDGRID_API_KEY‘, ‘app.frontend.base-url‘ | Used in rich email templates. |

> Need a walkthrough for Google OAuth? Follow ‘GOOGLE_OAUTH2_SETUP.md‘.

Database & Seed Data

- First-time startup automatically executes every SQL file in ‘backend/src/main/re
 \(schema creation, extensions, seed data\).
- To reset the data, drop/recreate the Postgres database and restart the backend \
- If you manage Postgres manually, create a dedicated database/user:
  ‘‘‘bash
  psql postgres
  CREATE DATABASE auctionality;
  CREATE USER auctionality WITH PASSWORD ’super-secret’;
  GRANT ALL PRIVILEGES ON DATABASE auctionality TO auctionality;
  ‘‘‘

Running & Testing

‘‘‘bash
cd backend
Compile & verify
./mvnw clean verify

Run with live reload + preview features enabled
./mvnw spring-boot:run -Dspring-boot.run.jvmArguments=--enable-preview

Package runnable jar
./mvnw -DskipTests package
java --enable-preview -jar target/auctionality-0.0.1-SNAPSHOT.jar
‘‘‘

- Backend listens on ‘http://localhost:8081‘.
- REST endpoints live under ‘/api/**‘.
- Swagger UI: ‘http://localhost:8081/swagger-ui.html‘.
- Actuator/health endpoints enabled at ‘/actuator/**‘ \(see ‘MONITORING_AND_LOGGIN

Messaging \(RabbitMQ\)

The backend declares:

Exchange	Queues	Routing keys	Purpose

| ‘bid.exchange‘ | ‘bid.history.queue‘, ‘product.price.queue‘ | ‘bid.history‘, ‘pr
| Broadcasts bid history updates and price changes to listeners \(SSE/WebSocket\).

Start RabbitMQ before running the backend; the default credentials ‘guest/guest‘ m
your environment variables.

Logging & Files

- Default log files: ‘logs/auctionality.log‘ and ‘logs/auctionality-error.log‘ \(r
- Adjust log settings in ‘backend/src/main/resources/logback-spring.xml‘.
- For monitoring/alerting guidance, read ‘MONITORING_AND_LOGGING.md‘.

Frontend \(React + Vite\)

Install & Run

‘‘‘bash
cd frontend
npm install
npm run dev   # starts http://localhost:5173
npm run build   # outputs to dist/
npm run preview   # serves the production build on http://localhost:4173
npm run lint   # ESLint checks
‘‘‘

Environment Variables

Create ‘frontend/.env.local‘ \(ignored by Git\) to decouple dev/prod URLs:

‘‘‘dotenv
VITE_API_BASE_URL=http://localhost:8081/api
VITE_API_URL=http://localhost:8081
VITE_WS_URL=http://localhost:8081/ws-chat
VITE_RECAPTCHA_SITE_KEY=your-site-key
‘‘‘

- ‘VITE_API_BASE_URL‘ drives Axios REST calls.
- ‘VITE_API_URL‘ is used by SSE helpers.
- ‘VITE_WS_URL‘ configures the WebSocket endpoint \(STOMP chat/bidding\).
- ‘VITE_RECAPTCHA_SITE_KEY‘ must match the backendâ s ‘RECAPTCHA_SITE_KEY‘.

Restart the Vite dev server whenever you change ‘.env.local‘.

Talking to the Backend

- The frontend assumes backend APIs are available on ‘http://localhost:8081‘. Upda
‘.env.local‘ file if you proxy or deploy elsewhere.
- OAuth redirect URLs point at the backend; ensure ‘FRONTEND_BASE_URL‘ matches you
frontend to complete the login redirect loop.

Monitoring & Observability

- Spring Boot Actuator is enabled by default. Check health via ‘curl
http://localhost:8081/actuator/health‘.
- Prometheus scrapes ‘http://localhost:8081/actuator/prometheus‘.
- Use the prebuilt stack:
  ‘‘‘bash
  docker-compose -f docker-compose.monitoring.yml up -d
  ‘‘‘
  - **Prometheus** â ‘http://localhost:9090‘
  - **Grafana** â ‘http://localhost:3000‘ \(admin/admin on first login\)
- See ‘MONITORING_SETUP.md‘ for provisioning details, dashboards, and troubleshoot

Full-Stack Workflow Checklist

1. **Start PostgreSQL & RabbitMQ** \(local install or Docker\).
2. **Set backend environment/secrets**.
3. **Run backend** \(‘./mvnw spring-boot:run -Dspring-boot.run.jvmArguments=--enab
4. **Populate frontend ‘.env.local‘**.
5. **Run frontend** \(‘npm run dev‘\) and verify ‘http://localhost:5173‘.
6. **Exercise features** â login/register \(with reCAPTCHA and optional Google O
products, test notifications.
7. **Monitor logs/metrics** via ‘logs/‘ and Grafana as needed.

Supporting Documentation

- ‘docs/Proposal.md‘ â business and functional overview.
- ‘GOOGLE_OAUTH2_SETUP.md‘ â end-to-end Google OAuth2 instructions.
- ‘MONITORING_SETUP.md‘ & ‘MONITORING_AND_LOGGING.md‘ â observability and loggin
material.

Troubleshooting

- **‘ClassNotFoundException‘ or preview warnings** â ensure you are using JDK 25
‘--enable-preview‘ when running the app or tests.
- **Database connection failures** â verify Postgres is listening on the host/po
your secrets and that the user has privileges. Use ‘psql -h localhost -U auctional
auctionality‘.
- **RabbitMQ errors** â confirm the service is running and credentials align wit

‘application.yml‘. Visit ‘http://localhost:15672‘ for the management UI.
- **OAuth redirect mismatches** â double-check redirect URIs in Google Cloud and ‘FRONTEND_BASE_URL‘.
- **reCAPTCHA failing locally** â Google provides global test keys shown in ‘secrets.properties‘; replace with production keys before deploying.
- **Frontend canâ t reach backend** â set ‘VITE_API_BASE_URL‘ and ‘VITE_API_URL run Vite with ‘npm run dev -- --host 0.0.0.0‘ if you access from another device.

With the steps above you can go from a bare clone to a fully working Auctionality minutes while keeping secrets, dependencies, and tooling organized. Happy building