Local Development Setup

Local Development Setup

> **Project:** {{PROJECT_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}}
> **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

---

1. System Requirements

Software	Required Version	macOS	Linux	Windows (WSL2)
{{RUNTIME}}	`{{VERSION}}`	`brew install {{RUNTIME}}`	`apt install {{RUNTIME}}`	Via WSL2
Docker Desktop	`{{DOCKER_VERSION}}+`	docker.com	docs	docs
Git	`2.30+`	Pre-installed	`apt install git`	Via WSL2
{{OTHER_TOOL}}	`{{VERSION}}`	`brew install {{OTHER_TOOL}}`	`{{LINUX_INSTALL}}`	{{WINDOWS_INSTALL}}

Hardware minimum: {{MIN_RAM}}GB RAM, {{MIN_DISK}}GB free disk space **Recommended:** {{REC_RAM}}GB RAM, SSD

Not supported: Windows without WSL2

---

2. Quick Start (5 Steps)

```
# 1. Clone
git clone {{REPO_URL}} && cd {{REPO_NAME}}

# 2. Install dependencies
{{INSTALL_CMD}}

# 3. Configure
cp .env.example .env
# Edit .env — see Section 5 for required values

# 4. Run (with Docker for dependencies)
{{START_CMD}}

# 5. Verify
open http://localhost:{{PORT}}
# Should see: {{EXPECTED_LANDING}}
```

Total time (fast path): ~{{QUICKSTART_TIME}} minutes

If anything fails, read the full setup in Section 3 or check Common Issues.

---

3. Detailed Setup

3.1 Runtime Version Manager

Tool: {{VERSION_MANAGER}}

```
# Install version manager
{{VM_INSTALL_CMD}}

# Install required runtime version
{{VM_USE_CMD}} {{RUNTIME_VERSION}}

# Set as default
{{VM_DEFAULT_CMD}} {{RUNTIME_VERSION}}

# Verify
{{RUNTIME}} --version  # Should output: {{RUNTIME_VERSION}}
```

Auto-switching: The project root contains `.{{VERSION_FILE}}` (e.g., `.nvmrc`, `.python-version`). If your shell is configured, it switches automatically on `cd` into the project.

3.2 Package Manager Setup

```
# The project uses {{PKG_MANAGER}}
# Install it if not available:
{{PKG_MANAGER_INSTALL}}

# Install project dependencies
cd {{REPO_NAME}}
{{INSTALL_CMD}}

# Verify installation
{{INSTALL_VERIFY_CMD}}
```

Lockfile: `{{LOCKFILE}}` — committed to git. Do NOT delete it. Run `{{INSTALL_CMD}}` (not `{{FORCE_INSTALL_CMD}}`) to respect lockfile.

3.3 Database Setup

Option A: Docker (recommended)

```
# Start database container
docker-compose up -d db

# Verify database is running
{{DB_CHECK_CMD}}

# Run migrations
{{MIGRATE_CMD}}

# Seed development data
{{SEED_CMD}}
```

Option B: Local database install

```
# Install PostgreSQL (example)
brew install postgresql@{{PG_VERSION}}  # macOS
# or: sudo apt install postgresql-{{PG_VERSION}}  # Linux
```

```
# Start database
{{DB_START_LOCAL_CMD}}

# Create database
createdb {{DB_NAME}}_dev

# Run migrations
{{MIGRATE_CMD}}

# Seed development data
{{SEED_CMD}}
```

Verify database is set up:

```
{{DB_VERIFY_CMD}}
# Should output: {{DB_VERIFY_EXPECTED}}
```

3.4 Environment Variables

```
# Copy the example file
cp .env.example .env
```

Edit `.env` with your local values:

Variable	Value for Local	Notes
`DATABASE_URL`	`{{LOCAL_DB_URL}}`	Update with your local DB creds
`REDIS_URL`	`redis://localhost:6379`	Default if using Docker
`PORT`	`{{PORT}}`	
`NODE_ENV`	`development`	
`{{SECRET_VAR}}`	—	Ask your onboarding buddy

Never commit `.env`: It's in `.gitignore`. Use `.env.example` for sharing template values.

3.5 SSL Certificates (If Needed Locally)

```
# Install mkcert
brew install mkcert    # macOS
# or: apt install mkcert  # Linux
```

```
# Install local CA
mkcert -install

# Generate cert for localhost
mkcert localhost 127.0.0.1 ::1

# Move certs to project (check .env for expected paths)
mv localhost+2.pem {{CERT_PATH}}
mv localhost+2-key.pem {{KEY_PATH}}
```

3.6 Seed Data Loading

```
# Load fixture data (fast, for unit/integration tests)
{{FIXTURE_CMD}}

# Load realistic development data (slower, ~{{SEED_SIZE}} records)
{{DEV_SEED_CMD}}

# Reset to clean state
{{RESET_CMD}}
```

Seeded accounts after running seed:

Role	Email	Password	Notes
Admin	`admin@dev.local`	`{{SEED_PWD}}`	Full access
User	`user@dev.local`	`{{SEED_PWD}}`	Standard user

---

4. Running the Application

Development Server

```
# Start development server (with hot reload)
{{DEV_CMD}}

# Application available at:
http://localhost:{{PORT}}
```

What starts: {{DEV_WHAT_STARTS}}

Watch Mode / Hot Reload

Hot reload is enabled by default in development. When you save a file:

- **Backend:** {{BACKEND_RELOAD}}
- **Frontend:** {{FRONTEND_RELOAD}}

Debug Mode

```
# Start with Node.js debugger
{{DEBUG_CMD}}

# Then attach your IDE debugger to port {{DEBUG_PORT}}
```

VS Code launch config: `.vscode/launch.json` — includes pre-configured debug targets.

Mobile Development (If Applicable)

```
# iOS Simulator (macOS only)
{{IOS_CMD}}

# Android Emulator
{{ANDROID_CMD}}
```

Requires: Xcode (iOS), Android Studio (Android)

---

5. Running Tests

Unit Tests

```
# Run all unit tests
{{UNIT_CMD}}

# Run tests in watch mode (re-runs on file save)
{{UNIT_WATCH_CMD}}
```

```
# Run specific test file
{{UNIT_CMD}} {{TEST_FILE_PATH}}

# Run tests matching pattern
{{UNIT_CMD}} --grep "{{PATTERN}}"
```

Integration Tests

```
# Requires database and Redis running (use Docker)
docker-compose up -d db redis

# Run integration tests
{{INT_CMD}}
```

E2E Tests

```
# Requires full application stack running
{{DEV_CMD}} &  # Or start in separate terminal

# Run E2E tests (headless)
{{E2E_CMD}}

# Run E2E with browser visible (debugging)
{{E2E_HEADED_CMD}}

# Run specific E2E test
{{E2E_CMD}} --grep "{{PATTERN}}"
```

Test Coverage Report

```
{{COV_CMD}}

# Report generated at: {{COV_REPORT_PATH}}
# Open in browser:
open {{COV_REPORT_PATH}}/index.html
```

---

6. Common Issues & Solutions

Problem	Likely Cause	Solution
`Module not found`	Dependencies not installed	Run `{{INSTALL_CMD}}`
`Database connection refused`	DB not running	Run `docker-compose up -d db`
`Port {{PORT}} already in use`	Another process using the port	`lsof -ti:{{PORT}} | xargs kill`
`Permission denied: .env`	File permission issue	`chmod 600 .env`
`Node version mismatch`	Wrong Node version active	`{{VERSION_MANAGER}} use {{NODE_VERSION}}`
Slow performance on Docker (macOS)	Docker Desktop volume mounts	Enable VirtioFS in Docker Desktop settings
`EACCES: permission denied, mkdir node_modules`	npm global prefix issue	See npm fix
Tests failing with timeout	Integration test DB not running	`docker-compose up -d`
`SSL: certificate verify failed`	Missing cert or mkcert not installed	Re-run mkcert setup (Section 3.5)
`Migration already applied`	Stale migration state	`{{MIGRATION_ROLLBACK_CMD}}` then re-run

Still stuck? Ask in `#{{DEV_CHANNEL}}` or ping your onboarding buddy.

---

7. Docker Setup (Alternative)

7.1 docker-compose.yml Reference

```
# Start all services
docker-compose up

# Start only backend services (db, redis, etc.)
docker-compose up db redis

# Rebuild after Dockerfile changes
docker-compose up --build

# Stop and remove containers
docker-compose down
```

```
# Stop and remove containers + volumes (reset state)
docker-compose down -v
```

Services defined in `docker-compose.yml`:

Service	Port	Description
`app`	`{{PORT}}`	Main application
`db`	`{{DB_PORT}}`	PostgreSQL database
`redis`	`6379`	Redis cache
`{{OTHER}}`	`{{PORT}}`	{{DESCRIPTION}}

7.2 Volume Mounts

Local Path	Container Path	Purpose
`.`	`/app`	Source code (hot reload)
`db-data`	`/var/lib/postgresql/data`	Database persistence
`node_modules`	`/app/node_modules`	Isolated from host

7.3 Port Mappings

Host Port	Container Port	Service
`{{PORT}}`	`{{PORT}}`	Application
`{{DB_PORT}}`	`5432`	PostgreSQL
`6379`	`6379`	Redis

---

8. IDE Configuration

8.1 Recommended Extensions (VS Code)

```
# Install all at once
cat .vscode/extensions.json | jq '.recommendations[]' | xargs -I{} code --install-extension {}
```

Extension	ID	Purpose
{{EXT_1}}	`{{ID}}`	{{PURPOSE}}

Extension	ID	Purpose
{{EXT_2}}	`{{ID}}`	{{PURPOSE}}
{{EXT_3}}	`{{ID}}`	{{PURPOSE}}
ESLint	`dbaeumer.vscode-eslint`	Inline linting
Prettier	`esbenp.prettier-vscode`	Format on save

8.2 Debug Configurations

The project includes `.vscode/launch.json` with these pre-built debug targets:

Configuration	What it does
`Debug: API Server`	Starts API with debugger attached on port {{DEBUG_PORT}}
`Debug: Tests`	Runs test suite with breakpoints supported
`Attach to Process`	Attach debugger to running process

8.3 Workspace Settings

`.vscode/settings.json` (committed to repo):

```
{
  "editor.formatOnSave": true,
  "editor.defaultFormatter": "{{FORMATTER_ID}}",
  "editor.codeActionsOnSave": {
    "source.fixAll.eslint": true
  },
  "typescript.tsdk": "node_modules/typescript/lib"
}
```

Related Documents

- Developer Onboarding Guide
- Coding Standards
- Environment Configuration

Approval

Role	Name	Date	Signature
Author			
Reviewer			
Approver			

Revision #4
Created 2026-02-24 14:54:34 UTC by John
Updated 2026-06-28 20:03:54 UTC by John