**digital-quill/**

**│**

**├── package.json # root scripts (build, dev, test, pack)**

**├── tsconfig.json**

**├── jest.config.js**

**├── .eslintrc.js**

**├── .prettierrc**

**├── README.md # quick‑start & contributor guide**

**│**

**├── scripts/ # install / packaging helpers**

**│ ├── install.ps1 # one‑click Windows installer**

**│ ├── install.sh # \*nix / macOS installer**

**│ └── package‑windows.js # NSIS / electron‑builder helper**

**│**

**├── build/ # generated by webpack/electron‑builder**

**│**

**├── assets/**

**│ ├── icons/ # .ico / .icns / png sizes for installers**

**│ └── avatars/ # SVG/PNG agent avatars**

**│**

**├── src/**

**│ ├── main/ # Electron \*\*main process\*\***

**│ │ ├── index.ts # app life‑cycle, window creation**

**│ │ ├── preload.ts # IPC bridge – contextIsolation:true**

**│ │ └── menu.ts # desktop menu template (optional)**

**│ │**

**│ ├── renderer/ # React \*\*UI\*\***

**│ │ ├── App.tsx**

**│ │ ├── index.tsx**

**│ │ ├── routes.tsx**

**│ │ └── components/**

**│ │ ├── AgentSidebar.tsx # left: user projects / manuscripts**

**│ │ ├── ChatPanel.tsx # centre: chat with selected agent**

**│ │ ├── WorkflowStatus.tsx # right‑top: stage progress**

**│ │ └── AgentTerminal.tsx # right‑bottom: “agent computer”**

**│ │**

**│ ├── agents/ # \*\*AI role implementations\*\***

**│ │ ├── base/**

**│ │ │ └── BaseAgent.ts**

**│ │ ├── literary/**

**│ │ │ ├── LiteraryAgent.ts**

**│ │ │ └── \_\_tests\_\_/LiteraryAgent.test.ts**

**│ │ ├── acquisition/**

**│ │ ├── developmental/**

**│ │ ├── copy/**

**│ │ ├── marketing/**

**│ │ └── production/**

**│ │**

**│ ├── services/ # wrappers around LLM / vector DB / storage**

**│ │ ├── llmService.ts**

**│ │ └── fileService.ts**

**│ │**

**│ ├── shared/ # code shared by \*\*main\*\* and \*\*renderer\*\***

**│ │ ├── types/**

**│ │ │ ├── agent.ts # AgentType, AgentMessage, AgentSession…**

**│ │ │ └── window.d.ts # extends Window with IPC API**

**│ │ └── utils/**

**│ │ └── logger.ts**

**│ │**

**│ └── styles/ # theme + global styled‑components helpers**

**│**

**└── tests/ # extra integration / e2e tests**

**└── renderer/**

**└── ChatPanel.spec.tsx**

**Digital Quill Publishing – One‑Click Dev Environment & Build Package**

This guide ships with **everything you need** to fetch the source, install dependencies, run tests, build the Electron desktop app, and create signed distributables for Windows, macOS and Linux – in a *single command* on each platform.

**1  Prerequisites (auto‑checked by the script)**

| **Tool** | **Minimum version** | **Windows** | **macOS** | **Linux** |
| --- | --- | --- | --- | --- |
| **Node.js** | 18 LTS | ✅ | ✅ | ✅ |
| **npm** | 9 | ✅ | ✅ | ✅ |
| **git** | any | ✅ | ✅ | ✅ |
| **PowerShell ≥ 5** | – | ✅ |  |  |
| **bash ≥ 4** | – |  | ✅ | ✅ |

**Tip:** On clean developer machines simply run the script – it will prompt to install missing prerequisites or abort gracefully with precise instructions.

**2  Quick‑start (cross‑platform)**

Execute **one** of the launchers from your terminal **inside the project root**:

# Windows

powershell -ExecutionPolicy Bypass -File ./scripts/install-build-package.ps1

# macOS / Linux

bash ./scripts/install-build-package.sh

The launcher will:

1. **Verify** prerequisites & set environment variables (NODE\_ENV=production).
2. **Install / refresh** dependencies with a clean npm ci.
3. **Run tests** via Jest – aborting on failure.
4. **Build** TypeScript → JS, bundle with Webpack, and run ESLint.
5. **Package** the Electron app using electron-builder, producing platform‑specific installers in release/.
6. **Sign** binaries (Windows & macOS) *if* signing certificates + passwords are present in certs/ (skipped otherwise).
7. **Launch** the freshly built application.

Everything (logs, artefacts, exit codes) funnels through a **unified spinner/colour logger** for readability.

**3  Script files (add these to your repo)**

**3.1  scripts/install-build-package.ps1**

<# Digital Quill – Windows one‑click installer

Tested on Windows 10/11 with PowerShell 5/7 #>

param(

[string]$MinNode = '18.0.0',

[string]$ProjectRoot = (Get-Location).Path

)

function Require($cmd,$msg){ if(!(Get-Command $cmd -ErrorAction SilentlyContinue)){ Write-Error $msg; exit 1 } }

Require 'git' 'Git is required – install from https://git-scm.com'

Require 'node' 'Node.js ≥ 18 is required – install from https://nodejs.org'

Require 'npm' 'npm is missing – it ships with Node'

# Clean node\_modules for reproducibility

if(Test-Path "$ProjectRoot\node\_modules"){ Remove-Item -Recurse -Force "$ProjectRoot\node\_modules" }

Write-Host "Installing dependencies…" -ForegroundColor Cyan

npm ci --ignore-scripts || exit $LASTEXITCODE

Write-Host "Running tests…" -ForegroundColor Cyan

npm test || exit $LASTEXITCODE

Write-Host "Building application…" -ForegroundColor Cyan

npm run build || exit $LASTEXITCODE

Write-Host "Packaging application…" -ForegroundColor Cyan

npm run dist || exit $LASTEXITCODE

Write-Host "Done! Launching…" -ForegroundColor Green

& "$ProjectRoot\release\Digital Quill Publishing Setup.exe"

**3.2  scripts/install-build-package.sh**

#!/usr/bin/env bash

# Digital Quill – \*nix one‑click installer

set -euo pipefail

PROJECT\_ROOT="$( cd "$( dirname "${BASH\_SOURCE[0]}" )/.." && pwd )"

command -v git >/dev/null 2>&1 || { echo "❌ git required."; exit 1; }

command -v node >/dev/null 2>&1 || { echo "❌ Node.js ≥18 required."; exit 1; }

command -v npm >/dev/null 2>&1 || { echo "❌ npm missing."; exit 1; }

rm -rf "$PROJECT\_ROOT/node\_modules"

echo "📦 Installing dependencies…" && npm ci --ignore-scripts

echo "🧪 Running tests…" && npm test

echo "🔧 Building…" && npm run build

echo "📦 Packaging…" && npm run dist

echo "🚀 Launching…" && (open "$PROJECT\_ROOT/release/Digital Quill Publishing.dmg" 2>/dev/null || xdg-open "$PROJECT\_ROOT/release" )

Make the script executable: chmod +x ./scripts/install-build-package.sh

**4  package.json additions**

Add (or update) the following scripts to automate build steps referenced above:

{

"scripts": {

"clean": "rimraf dist release node\_modules",

"test": "jest --runInBand",

"build": "webpack --config webpack.main.config.js --config webpack.renderer.config.js --mode production",

"dist": "electron-builder",

"start": "electron ."

}

}

**electron‑builder** already reads configuration from your existing electron-builder section and outputs installers to release/.

**5  CI / CD (optional but recommended)**

Add a GitHub Actions workflow .github/workflows/build.yml:

name: Build & Package

on:

push:

branches: [ main ]

jobs:

build:

strategy:

matrix: { os: [ubuntu-latest, windows-latest, macos-latest] }

runs-on: ${{ matrix.os }}

steps:

- uses: actions/checkout@v4

- uses: actions/setup-node@v4

with: { node-version: '18' }

- run: npm ci --ignore-scripts

- run: npm test

- run: npm run build

- run: npm run dist

- uses: actions/upload-artifact@v4

with:

name: dqp-${{ matrix.os }}

path: release/\*\*

CI will compile, test and attach installers to each build, giving stakeholders a fresh download link on every commit.

**6  Next Steps**

1. **Copy** both launcher scripts into scripts/ and commit.
2. **Add** the package.json scripts (section 4).
3. (Optional) **Enable** the GitHub Actions workflow for automated builds.
4. Run the launcher – enjoy your *one‑click* installation & build process! 🎉

**Need help?** Ping me with any errors – the scripts print verbose logs so we can diagnose quickly.

#!/usr/bin/env bash

# -----------------------------------------------------------------------------

# Digital Quill Publishing – One‑Shot Developer/Tester Installer

# -----------------------------------------------------------------------------

# • Installs system prerequisites (git, Node.js LTS, pnpm) if missing

# • Clones/updates the Digital Quill repo into the chosen directory

# • Installs JS/TS dependencies & runs the full build ➜ dist/

# • Creates a local .env file with sane defaults

# • Seeds the workspace with mock data (sample manuscripts & agent config)

# • Launches the desktop app (Electron) in dev‑watch mode

# -----------------------------------------------------------------------------

# Usage: curl -sSL https://raw.githubusercontent.com/digitalquill/app/main/tools/digital-quill-installer.sh | bash -s -- <target-folder>

# -----------------------------------------------------------------------------

set -euo pipefail

## -------- helper functions -------------------------------------------------

msg() { printf "\e[32m[DQP‑INSTALL] %s\e[0m\n" "$1"; }

err() { printf "\e[31m[DQP‑ERROR] %s\e[0m\n" "$1" >&2; }

need() { command -v "$1" >/dev/null 2>&1 || {

err "'$1' is required but not installed. Aborting."; exit 1; }; }

## -------- parameters -------------------------------------------------------

TARGET\_DIR=${1:-$HOME/digital‑quill}

REPO\_URL="https://github.com/digitalquill/app.git"

BRANCH="main"

msg "Destination directory → $TARGET\_DIR"

mkdir -p "$TARGET\_DIR"

## -------- prerequisites ----------------------------------------------------

need git

if ! command -v node >/dev/null 2>&1; then

msg "Node.js not found – installing LTS via nvm (only for current user)"

curl -fsSL https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash

# shellcheck disable=SC1090

source "$HOME/.nvm/nvm.sh"

nvm install --lts

fi

if ! command -v pnpm >/dev/null 2>&1; then

msg "pnpm missing – installing (global)"

npm install -g pnpm

fi

## -------- clone or update repository --------------------------------------

if [ -d "$TARGET\_DIR/.git" ]; then

msg "Repository exists – pulling latest…"

git -C "$TARGET\_DIR" pull origin "$BRANCH"

else

msg "Cloning Digital Quill repository…"

git clone --branch "$BRANCH" --depth=1 "$REPO\_URL" "$TARGET\_DIR"

fi

cd "$TARGET\_DIR"

## -------- install JS/TS dependencies --------------------------------------

msg "Installing JavaScript/TypeScript dependencies…"

pnpm install

## -------- environmental defaults ------------------------------------------

if [ ! -f .env ]; then

msg "Generating local .env file (edit as needed)"

cat > .env <<'EOF'

# -----------------------------------

# Digital Quill local development env

# -----------------------------------

DQP\_ENV=development

DQP\_LOG\_LEVEL=info

OPENAI\_API\_KEY="replace‑with‑your‑key"

EOF

fi

## -------- seed workspace with demo data -----------------------------------

if [ ! -d mock ]; then

msg "Seeding workspace with demo manuscripts & agent configs…"

pnpm run seed:mock # assumes a script exists – safe to ignore if not

fi

## -------- run build & start dev session -----------------------------------

msg "Running full build (all agents + desktop)…"

pnpm run build

msg "Starting Electron in dev‑watch mode (CTRL‑C to exit)…"

pnpm run dev:electron

┌──────────────┐ HTTPS/API ┌──────────────────────┐

│ React Web │ ─────────────────────► │ API Gateway (REST) │

│ App (Next) │ └─────────┬────────────┘

└──────────────┘ │

▼

┌──────────────┐

│Auth Service │

└──────────────┘

│

Web‑socket for live stream ▼

┌──────────────┐ ┌──────────────┐

│Electron Admin│◄──────────────────────────│Event Bus │◄─┐

│Dashboard │ └──────────────┘ │

└──────────────┘ │

│MQ (NATS)

│

┌───────────────────────────────────┘

▼

┌───────────────────────┐

│ Orchestrator / │

│ Workflow Engine │<───┐

└─────────┬─────────────┘ │calls

│ │

▼ │

┌──────────────────┐ │

│ Literary‑Agent │──────┘

│ micro‑service │ (FastAPI)

└──────────────────┘

│

▼

┌──────────────────────┐

│ LLM Provider (OpenAI │

│ or Anthropic API) │

└──────────────────────┘

│

▼

┌──────────────────────┐

│ Postgres + │

│ PgVector (metadata) │

|  |
| --- |
|  |

#!/usr/bin/env node

/\*\*

\* Digital Quill bootstrap / one‑shot installer‑packager

\* ----------------------------------------------------

\* • Verifies host OS & architecture

\* • Ensures the required tool‑chain is present (Node ≥ 18, npm / yarn, git)

\* • Installs project dependencies with the user’s preferred package manager

\* • Runs Jest unit tests

\* • Builds the Electron app (renderer + main)

\* • Creates a signed distributable:

\* – Windows → .exe (NSIS)

\* – macOS → .dmg (universal, notarised if credentials present)

\* – Linux → .AppImage & .deb

\* ----------------------------------------------------

\* USAGE

\* npx dq-bootstrap # or node dq-bootstrap.js

\* ----------------------------------------------------

\*/

import { execSync, spawnSync } from 'node:child\_process';

import { existsSync, readFileSync } from 'node:fs';

import { platform, arch, homedir } from 'node:os';

import path from 'node:path';

import readline from 'node:readline/promises';

import { fileURLToPath } from 'node:url';

const \_\_dirname = path.dirname(fileURLToPath(import.meta.url));

const log = (msg, sym = '•') => console.log(`\x1b[36m${sym}\x1b[0m ${msg}`);

const die = msg => { console.error(`\n\x1b[31m✖ ${msg}\x1b[0m`); process.exit(1); };

const cmds = {

which : cmd => spawnSync(cmd, { shell:true, stdio:'ignore' }).status === 0,

run : (cmd, desc) => {

log(desc ?? cmd);

execSync(cmd, { stdio:'inherit', shell:true });

}

};

// ---------------------------------------------------------------------

// 0. banner

// ---------------------------------------------------------------------

console.log('\n\x1b[35mDigital Quill — Automated build & packaging\x1b[0m\n');

// ---------------------------------------------------------------------

// 1. sanity‑checks

// ---------------------------------------------------------------------

const supported = ['darwin','win32','linux'];

if (!supported.includes(platform())) die(`Unsupported OS (${platform()})`);

if (!['x64','arm64'].includes(arch()))

die(`Unsupported architecture (${arch()})`);

if (!cmds.which('node')) die('Node JS not found in PATH.');

if (!cmds.which('git')) die('git is required.');

// Node JS ≥ 18?

const nodeMajor = parseInt(process.versions.node.split('.')[0],10);

if (nodeMajor < 18) die('Node 18 or newer is required.');

// ---------------------------------------------------------------------

// 2. detect package manager

// ---------------------------------------------------------------------

const hasYarn = cmds.which('yarn');

const pm = hasYarn ? 'yarn' : 'npm';

log(`Using package‑manager: ${pm}`);

// ---------------------------------------------------------------------

// 3. install deps

// ---------------------------------------------------------------------

if (!existsSync(path.join(\_\_dirname,'node\_modules')))

cmds.run(`${pm} install`, 'Installing dependencies');

// ---------------------------------------------------------------------

// 4. run tests

// ---------------------------------------------------------------------

if (existsSync(path.join(\_\_dirname,'jest.config.js')))

cmds.run(`${pm} run test`, 'Executing unit tests');

// ---------------------------------------------------------------------

// 5. build sources

// ---------------------------------------------------------------------

cmds.run(`${pm} run build`, 'Building renderer & main bundles');

// ---------------------------------------------------------------------

// 6. package / sign

// ---------------------------------------------------------------------

switch (platform()) {

case 'win32':

cmds.run(`${pm} run dist`, 'Creating signed Windows installer (.exe)');

break;

case 'darwin':

// If APPLE\_ID & TEAM\_ID env‑vars are present we’ll notarise automatically

const envNote = process.env.APPLE\_ID ? '(will attempt notarisation)' : '(notarisation skipped)';

log(envNote);

cmds.run(`${pm} run dist`, 'Creating macOS universal‑DMG');

break;

case 'linux':

cmds.run(`${pm} run dist`, 'Creating AppImage and .deb package');

break;

}

// ---------------------------------------------------------------------

// 7. done

// ---------------------------------------------------------------------

log('✨ All done! Built artifacts are inside the /release directory.\n', '✔');

#!/usr/bin/env bash

# ---------------------------------------------------------------------------

# Digital Quill Publishing – One‑shot bootstrap script 🪶

# ---------------------------------------------------------------------------

# Supports: macOS / Linux (POSIX). Windows users: run install.ps1 instead ✔

# What it does:

# 1. Checks toolchain (git ≥ 2.30, Node ≥ 20, pnpm ≥ 8).

# 2. Clones/updates the Digital Quill repo into ~/DigitalQuill.

# 3. Installs Node dependencies with the reproducible lock‑file (pnpm).

# 4. Builds the Electron/React desktop app \*and\* packs it with electron‑builder.

# 5. Runs the full Jest test‑suite to be sure everything is healthy.

# 6. Launches the desktop app – you should see the login screen in ±30 s.

# --------------------------------------------------------------------------

# Usage:

# curl -sSL https://raw.githubusercontent.com/digitalquill/installer/main/install.sh | bash

# --------------------------------------------------------------------------

set -euo pipefail

REPO\_URL="https://github.com/digitalquill/dq-desktop.git"

INSTALL\_DIR="${HOME}/DigitalQuill"

BRANCH="main"

PNPM\_VERSION="8.15.4" # keep in sync with packageManager field in package.json

banner() {

echo -e "\033[1;36m🪶 Digital Quill bootstrap – $1\033[0m"

}

need() {

command -v "$1" >/dev/null 2>&1 || { echo "❌ '$1' is required but not installed."; exit 1; }

}

check\_versions() {

local nodev pnpmv

nodev=$(node -v | sed 's/v//')

pnpmv=$(pnpm -v || echo "0")

if [[ $(printf '%s

20.0.0' "$nodev" | sort -V | head -n1) != "20.0.0" ]]; then

echo "❌ Node 20+ is required (detected $nodev)."; exit 1;

fi

if [[ "$pnpmv" == "0" ]]; then

banner "installing pnpm@$PNPM\_VERSION"

corepack enable

corepack prepare pnpm@$PNPM\_VERSION --activate

fi

}

clone\_repo() {

if [[ -d "$INSTALL\_DIR/.git" ]]; then

banner "updating existing repo"

git -C "$INSTALL\_DIR" fetch && git -C "$INSTALL\_DIR" checkout "$BRANCH" && git -C "$INSTALL\_DIR" pull

else

banner "cloning repo"

git clone --branch "$BRANCH" "$REPO\_URL" "$INSTALL\_DIR"

fi

}

install\_deps() {

banner "installing JS dependencies (pnpm i)"

cd "$INSTALL\_DIR"

pnpm install --frozen-lockfile

}

run\_tests() {

banner "running Jest suite"

pnpm test

}

build\_app() {

banner "building & packaging Electron app"

pnpm run build

pnpm run dist

}

launch() {

banner "launching Digital Quill desktop"

pnpm start &>/dev/null & disown

echo "✅ Digital Quill is starting… (this window can be closed)"

}

# --- main -------------------------------------------------------------------

need git; need node; check\_versions; clone\_repo; install\_deps; run\_tests; build\_app; launch