Installation Guide

Prerequisites

- Node.js v18+ and npm
- Python 3.9+
- Docker Desktop (recommended) with Docker Compose v2
- PostgreSQL (only required if running backend locally without Docker)

Quick Start (Docker Recommended)

Production-like Stack

```
# From project root
docker compose up -d --build
# Or legacy Compose v1
# docker-compose up -d --build

# Frontend: http://localhost:3000
# Backend API: http://localhost:5001
```

Dev Profile (hot reload for frontend)

```
docker compose --profile dev up -d --build
# Or legacy Compose v1
# docker-compose --profile dev up -d --build

# Frontend: http://localhost:3000
# Backend API: http://localhost:5001
```

Local Development (without Docker)

If you prefer to run the backend locally via `python app.py` and the frontend with `npm start`:

1) Start PostgreSQL

- Option A: Use the Compose-provisioned Postgres only

```
docker compose up -d db
# DB exposed on host: localhost:5433
```

- Option B: Use your own Postgres at localhost:5432

Create the database `test-project`:

```
# createdb CLI (choose the matching port)
createdb -h localhost -p 5433 -U postgres test-project  # Docker DB
# OR
createdb -h localhost -p 5432 -U postgres test-project  # Native Postgres

# psql alternative
psql -h localhost -p 5433 -U postgres -c "CREATE DATABASE \"test-project\";"
```

2) Backend setup

```
cd backend
python -m venv venv
# Windows: venv\Scripts\activate
# macOS/Linux: source venv/bin/activate
pip install -r requirements.txt

# Create backend/.env
```

Create `backend/.env` with:

```
# If using Compose Postgres on 5433
DATABASE_URL=postgresql://postgres:123456@localhost:5433/test-project
# If using native Postgres on 5432, adjust:
# DATABASE_URL=postgresql://postgres:YOUR_PASSWORD@localhost:5432/test-project

FLASK_ENV=development
FLASK_DEBUG=True
SECRET_KEY=dev-secret-key
```

Run the backend:

```
python app.py
# Backend: http://localhost:5001
```

3) Frontend setup

```
# From project root
npm install
npm start
# Frontend: http://localhost:3000
```

If you need a custom backend URL for the frontend, set `REACT_APP_BACKEND_URL` before build/start (Compose does this for you in Docker):

```
# macOS/Linux
env REACT_APP_BACKEND_URL=http://localhost:5001 npm start
# Windows (PowerShell)
$env:REACT_APP_BACKEND_URL="http://localhost:5001"; npm start
```

Testing

Frontend Component Tests (Cypress)

```
# Install dependencies
npm install

# Headless with coverage (recommended)
npm run ct:cov

# Interactive runner (GUI)
npx cypress open --component

# Headless without coverage (faster)
npm run cypress:run:ct
```

Coverage report path:

- Generated at `coverage/ct/index.html`
- On Windows, it opens automatically after `npm run ct:cov`
- macOS: `open coverage/ct/index.html`
- Linux: `xdg-open coverage/ct/index.html`

Frontend E2E Tests (Cypress)

Ensure the app is running (Docker recommended) then:

```
 # Open interactive E2E runner
npm run cypress:open


### Backend Tests (pytest)

```bash
cd backend
# Activate venv if not already
pytest -q

# With coverage report
pytest --cov=. --cov-report=html
# HTML coverage: backend/htmlcov/index.html
```

Environment and Configuration

- Backend environment variables (via `backend/.env` or Compose):
  - `DATABASE_URL` : PostgreSQL connection string
  - `FLASK_ENV`, `FLASK_DEBUG`, `SECRET_KEY`
- Frontend backend URL: `REACT_APP_BACKEND_URL` (handled by Compose for Docker workflows)
- CORS: Preconfigured in `backend/app.py` to allow `http://localhost:3000` and `http://localhost:3001`

Common Commands

```
 # Rebuild and restart everything (Docker v2)
docker compose up -d --build

# Start only the database
docker compose up -d db

# Rebuild only backend
docker compose up -d --build backend

# Stop and remove containers
docker compose down
```

Troubleshooting

- Coverage flake in Cypress CT: run headless (`npm run ct:cov`) or temporarily disable code coverage in `test/frontend/support/component.js` .
- Port conflicts: Ensure nothing else listens on 3000/5001/5433.
- Database errors: Confirm `DATABASE_URL` and the `test-project` database exist; recreate with the commands above.
```