# Setup (No Docker)

Okay, I have performed a final, comprehensive verification and confirmation of my understanding based on all information exchanged, including the original assignment documents, evaluation criteria, all proposals, critiques, and the explicit instruction to **exclude everything Docker-related**.

This is the definitive, finalized proposal for the Bookworm development environment setup, constructed from scratch with an **absolutely minimalist mindset** and approach, tailored for an individual rookie assignment running directly on the local machine.

**1. All Development Tools, Libraries, and Frameworks**

- **Rationale:** This list comprises the essential technologies required by the assignment's stack (React, FastAPI, PostgreSQL), assuming direct local installation. It uses current, stable versions and excludes containerization tools and optional helpers to maintain minimalism.
- **Final List:**
  - **Languages:** Python (**v3.11+** recommended, installed locally), JavaScript (ES6+)
  - **Backend Framework:** FastAPI

- **Frontend Library:** React (**v18.3+**)
- **Database:** PostgreSQL (**v16+** recommended, installed and running locally)
- **Styling:** Bootstrap 5 CSS (**v5.3.x**, Direct class usage only)
- **API Server (Python):** Uvicorn (Standard ASGI server for FastAPI)
- **Database Driver (Python):** `psycopg2-binary` (Chosen for simpler local installation; *Note:* `psycopg` *(v3) is the modern alternative offering better performance/features if installation is not an issue*).
- **ORM (Python):** SQLAlchemy (**v2.0+ syntax** recommended)
- **Data Validation (Python):** Pydantic (**v2+ usage** recommended)
- **Auth Libraries (Python):** `python-jose[cryptography]` (JWT), `passlib[bcrypt]` (Hashing)
- **API Client (Frontend):** Native Browser `fetch` API (Built-in, minimalist)
- **Routing (Frontend):** `react-router-dom` (**v6+**)
- **Build Tool (Frontend):** Vite (**v5+**)
- **Package Managers:** `pip` (Python), `npm` (Node.js)
- **Version Control:** Git
- **Python Virtual Environment:** `venv` (Built into Python 3)
- **Code Editor:** Any capable editor (VS Code recommended)
- **Database GUI (Optional):** pgAdmin, DBeaver, or similar (for local DB interaction)

## 2. Essential Packages/Libraries to Include

- **Rationale:** Listing only the direct, installable dependencies needed for core functionality based on the chosen tools for local execution. Uses the simpler synchronous DB driver ( `psycopg2-binary` ) for initial setup ease. Exact versions should be pinned after setup for reproducibility.
- **Backend ( `requirements.txt` - Initial):**

```
# Core framework & server
fastapi
uvicorn[standard]

# Database interaction
sqlalchemy>=2.0
psycopg2-binary # Sync driver for local setup simplicity. Consider
'psycopg>=3' as modern alternative.

# Data validation & Auth
pydantic[email]>=2.0
python-jose[cryptography]
passlib[bcrypt]

# Database migrations
alembic

# Dev environment helper
python-dotenv
```

*(Note: Generate exact versions using `pip freeze > requirements.txt` within the activated venv after initial install).*

- **Frontend ( `package.json` - dependencies):**

```
{
  "dependencies": {
    "react": "^18.3.1", // Example: Use `npm install react@latest` initially
    "react-dom": "^18.3.1", // Example: Use `npm install react-dom@latest` initially
    "react-router-dom": "^6.23.1", // Example: Use `npm install react-router-dom@latest` initially
    "bootstrap": "^5.3.3" // Example: Use `npm install bootstrap@latest` initially
  },
  "devDependencies": {
    "@vitejs/plugin-react": "^4.3.1", // Example: Use `npm install @vitejs/plugin-react@latest --save-dev` initially
    "vite": "^5.2.11" // Example: Use `npm install vite@latest --save-dev` initially
  }
}
```

*(Note: Versions are examples. Run `npm install <package>@latest` for each at project start to get current stable versions. Commit the resulting `package-lock.json`).*

### 3. Entire Project Structure

- **Rationale:** A standard monorepo structure provides the clearest organization for related backend and frontend codebases when Docker is not used, keeping everything in one repository but logically separated. Includes placeholders for local environment artifacts ignored by Git.
- **Final Structure:**

```
bookworm-project/
├── backend/
│   ├── app/                    # Main application source
│   │   ├── __init__.py
│   │   ├── main.py             # FastAPI app, CORS, routers
│   │   ├── api/                # API endpoints & dependencies
│   │   │   └── v1/
│   │   │       ├── __init__.py
│   │   │       ├── endpoints/
│   │   │       └── deps.py
│   │   ├── core/               # Config
│   │   ├── db/                 # DB Session setup
│   │   ├── models/             # SQLAlchemy models
│   │   ├── schemas/            # Pydantic schemas
│   │   └── services/           # Business logic
│   ├── alembic/                # Migration scripts
│   ├── alembic.ini             # Alembic config
│   ├── requirements.txt        # Python dependencies (pinned)
│   ├── .env                    # Local environment variables
(gitignored)
```

```
│   └── .env.example          # Template for .env
│   └── venv/                 # Python virtual environment
(gitignored)
├── frontend/
│   ├── public/               # Static assets
│   ├── src/                  # Frontend source code
│   │   ├── App.jsx
│   │   ├── main.jsx
│   │   ├── index.css         # Global styles, Bootstrap import
│   │   ├── api/
│   │   ├── components/
│   │   ├── contexts/
│   │   ├── hooks/
│   │   └── pages/
│   ├── node_modules/         # Node dependencies (gitignored)
│   ├── package.json
│   ├── package-lock.json     # Exact Node dependencies lockfile
│   ├── vite.config.js
│   ├── .env
│   └── .env.example
├── .gitignore                # Should ignore venv/,
node_modules/, .env* etc.

└── README.md                    # Setup instructions
```

**4. Set Up Development Environment Process**

- **Rationale:** This outlines the necessary manual steps for a local setup, emphasizing local installations and manual database creation, reflecting the exclusion of Docker.
- **Final Process:**
    1. **Install Prerequisites:** Install Git, Python (v3.11+), Node.js (**v22.x LTS or newer LTS** recommended), PostgreSQL (v16+ recommended). Ensure the PostgreSQL server service is running locally.
    2. **Create Local Database:** Using `psql` or a GUI tool (like pgAdmin, DBeaver), connect to your local PostgreSQL instance. Create a database (e.g., `bookworm_db`), a user/role (e.g., `bookworm_user`) with a secure password. Grant necessary privileges (CONNECT, SELECT, INSERT, UPDATE, DELETE, etc.) on the database to the user. Note these credentials.
    3. **Clone Project:** `git clone <repository_url> bookworm-project && cd bookworm-project`
    4. **Set Up Backend:**
        - `cd backend`

  - `python -m venv venv`
  - Activate venv (`source venv/bin/activate` or `.\venv\Scripts\activate`)
  - `pip install -r requirements.txt` (Installs packages listed initially)
  - Copy `.env.example` to `.env`. Edit `.env`: Set `DATABASE_URL=postgresql+psycopg2://bookworm_user:your_password@localhost:5432/bookworm_db` (use credentials from step 2; adjust driver in URL if using `psycopg`) and generate/set a strong `SECRET_KEY` (e.g., `python -c 'import secrets; print(secrets.token_hex(32))'`).

5. **Database Migrations (Initial):**
  - Ensure backend venv is active.
  - If `alembic/` doesn't exist: `alembic init alembic`.
  - Edit `alembic.ini`: Set `sqlalchemy.url = %(DATABASE_URL)s` (or paste the full URL).
  - Edit `alembic/env.py`: Update `target_metadata = None` to import your SQLAlchemy `Base` and all defined models (e.g., `from app.db.session import Base; from app.models import *; target_metadata = Base.metadata`).

- Generate initial migration: `alembic revision --autogenerate -m "Initial schema"`. **Review the generated script carefully.**
- Apply migration: `alembic upgrade head`. Verify tables exist in the local DB.

6. **Set Up Frontend:**
    - `cd ../frontend`
    - `npm install` (Installs packages based on `package.json`, creates `package-lock.json`). Commit `package-lock.json`.
    - Copy `.env.example` to `.env`. Edit `.env`: Set `VITE_API_BASE_URL` (e.g., `http://localhost:8000/api/v1`, matching the backend port and API prefix).
7. **Run Application:**
    - **Terminal 1 (Backend):** `cd backend`, activate venv, `uvicorn app.main:app --reload --port 8000`. Keep this running.
    - **Terminal 2 (Frontend):** `cd frontend`, `npm run dev`. Note the local URL provided (e.g., `http://localhost:5173`). Keep this running.
8. **Access:** Open the frontend URL in your browser.

## 5. Set Up Project Set Up Commands (and Configurations Options to Consider)

- **Rationale:** Listing essential commands for the local workflow and the minimal critical configurations needed to get started.
- **Final Commands & Config:**
  - **Backend (in `backend/` with active `venv`):** `source venv/bin/activate` (or `.\venv\Scripts\activate`), `pip install -r requirements.txt`, `uvicorn app.main:app --reload --port 8000`, `alembic revision --autogenerate -m "..."`, `alembic upgrade head`.
  - **Frontend (in `frontend/`):** `npm install`, `npm run dev`, `npm run build`.
  - **Minimal `.env` Variables:**
    - `backend/.env`: `DATABASE_URL` (pointing to `localhost` with correct credentials), `SECRET_KEY`.
    - `frontend/.env`: `VITE_API_BASE_URL` (pointing to `http://localhost:8000/...`).

### 6. Configuration Files with Best Practices for a Production Environment

- **Rationale:** Providing concise, high-level *guidance* relevant to a non-containerized deployment, focusing on security and operational concerns without prescribing specific files, aligning with the minimalist focus on the development setup.
- **Final Guidance:**

- **Server:** Provision server (VM/PaaS). Install required runtimes (Python, Node.js), PostgreSQL (or use managed DB), and a web server (Nginx recommended).
- **Deployment:** Use Git to deploy code. Set up backend Python `venv`. Install dependencies from pinned `requirements.txt`. Place frontend build artifacts (`frontend/dist/`) for Nginx to serve.
- **Configuration:** Use **server environment variables** for ALL secrets and configurations (`DATABASE_URL`, `SECRET_KEY`, API keys, etc.). Do not deploy `.env` files with production secrets.
- **Process Management:** Run the FastAPI application using `gunicorn` + `uvicorn` workers, managed by a robust process manager like `systemd` for reliability.
- **Web Server:** Configure Nginx (or similar) to serve static frontend files, reverse proxy API requests to the Gunicorn process, handle **HTTPS termination** (e.g., using Let's Encrypt), and implement essential security headers (CSP, HSTS, etc.).

## 7. Guidance on Dependencies Management

- **Rationale:** Reproducibility and security are fundamental development practices, even in a minimal setup. The guidance focuses on standard local environment practices.

- **Final Guidance:**
    - **Pinning:** Always commit `frontend/package-lock.json`. Generate and commit `backend/requirements.txt` with exact versions (`==`) using `pip freeze > backend/requirements.txt` **within the activated backend venv**.
    - **Isolation (Python):** Always use the activated `venv` for backend Python tasks (installing, running server, running migrations) to prevent conflicts.
    - **Auditing:** Regularly check for vulnerabilities using `npm audit` (in `frontend/`) and `pip-audit` (install with `pip install pip-audit` within the backend venv).
    - **Updating:** Update dependencies methodically in separate branches, test thoroughly, and commit the updated lockfile / `requirements.txt`.

This finalized proposal provides the leanest, most current, and essential development environment setup required to successfully start and complete the Bookworm rookie assignment *without Docker*, based on all provided specifications and adhering strictly to minimalist principles.