# Windows -

🚀

## 1. Node.js

1. https://nodejs.org/
2. **"Download Node.js (LTS)"** LTS
3. `.msi`
4.
   - **"Next"**
   - **"Next"**
   - **"Next"**
   - **"Add to PATH"** **"Next"**
   - **"Install"**
   - **"Finish"**

1. `Win + R` `cmd` Enter
2.
```cmd
node --version
npm --version
```
3.
```
v18.18.0
9.8.1
```

## 2. PostgreSQL

1. https://www.postgresql.org/download/windows/
2. **"Download the installer"**

3. PostgreSQL 15.x 16.x
4. `.exe`
5.
   - **"Next"**
   - **"Next"**
   - **"Next"**
   - **"Next"**
   - (postgres)
   - 5432 **"Next"**
   - **"Next"**
   - **"Next" → "Next" → "Finish"**

1. **"SQL Shell (psql)"**
2. Enter
3.
4. `postgres=#`

## 3. Visual Studio Code

1. https://code.visualstudio.com/
2. **"Download for Windows"**
3. `.exe`
4.
   - **"Next"**
   - **"Next"**
   -
     - ✅ Add "Open with Code" action to Windows Explorer file context menu
     - ✅ Add "Open with Code" action to Windows Explorer directory context menu
     - ✅ Register Code as an editor for supported file types
     - ✅ Add to PATH
   - **"Next" → "Install" → "Finish"**

**VS Code**

1. VS Code
2. 
3. 
   - **ES7+ React/Redux/React-Native snippets**
   - **Prettier - Code formatter**
   - **ESLint**
   - **Prisma**
   - **TypeScript Importer**

### 4. Git

1. https://git-scm.com/download/win
2. 
3. `.exe`
4. "Next"

```cmd
git --version
```

🏗️

### 1.

1. `my-projects`
2. → **"Open with Code"** VS Code

### 2.

VS Code

1. `Ctrl + Shift + ``
2. **Terminal → New Terminal**

## 3. Next.js

```bash
npx create-next-app@latest system-dev-management --typescript --tailwind --eslint --app --src-dir --import-ali
```

- 
- 
- `create-next-app` `y` Enter

## 4.

```bash
cd system-dev-management
```

## 5.

```bash
#
npm install prisma @prisma/client

#
npm install next-auth @auth/prisma-adapter

#
npm install bcryptjs
npm install --save-dev @types/bcryptjs

#
npm install clsx tailwind-merge lucide-react

#
npm install react-hook-form @hookform/resolvers zod

#
npm install --save-dev tsx
```

🗄️

## 1.

1. **"SQL Shell (psql)"**
2. Enter
3. PostgreSQL
4. `postgres=#`

```sql
CREATE DATABASE system_dev_management;
```

5. Enter `CREATE DATABASE`
6. `\q`

## 2.

`.env.local`

1. VS Code → **New File**
2. `.env.local`
3. **`your_password`**

```bash
# URL
DATABASE_URL="postgresql://postgres:your_password@localhost:5432/system_dev_management"

# NextAuth.js
NEXTAUTH_SECRET="your-super-secret-key-here-change-this-in-production"
NEXTAUTH_URL="http://localhost:3000"

#
NODE_ENV="development"
```

## 3. Prisma

```bash
```

```
#           Prisma
npx prisma init
```

## 4.         Prisma Schema

1.         `prisma/schema.prisma`
2.

```
prisma
```

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

// NextAuth.js
model Account {
  id                String  @id @default(cuid())
  userId            String
  type              String
  provider          String
  providerAccountId String
  refresh_token     String? @db.Text
  access_token      String? @db.Text
  expires_at        Int?
  token_type        String?
  scope             String?
  id_token          String? @db.Text
  session_state     String?

  user User @relation(fields: [userId], references: [id], onDelete: Cascade)

  @@unique([provider, providerAccountId])
}

model Session {
  id           String   @id @default(cuid())
  sessionToken String   @unique
  userId       String
  expires      DateTime
  user         User     @relation(fields: [userId], references: [id], onDelete: Cascade)
}

model VerificationToken {
  identifier String
  token      String   @unique
  expires    DateTime

  @@unique([identifier, token])
```

```
}

//
model User {
  id            String    @id @default(cuid())
  name          String?
  email         String    @unique
  emailVerified DateTime?
  image         String?
  password      String?
  role          String    @default("user")
  createdAt     DateTime  @default(now())
  updatedAt     DateTime  @updatedAt

  accounts Account[]
  sessions Session[]

  createdDictionaries DataDictionary[] @relation("DictionaryCreator")
  updatedDictionaries DataDictionary[] @relation("DictionaryUpdater")
  createdKnowledge    KnowledgeBase[]  @relation("KnowledgeCreator")
  updatedKnowledge    KnowledgeBase[]  @relation("KnowledgeUpdater")
  createdSystems      ApiSystem[]      @relation("SystemCreator")
  updatedSystems      ApiSystem[]      @relation("SystemUpdater")
  createdCategories   FunctionCategory[] @relation("CategoryCreator")
  updatedCategories   FunctionCategory[] @relation("CategoryUpdater")
  createdComponents   ApiComponent[]   @relation("ComponentCreator")
  updatedComponents   ApiComponent[]   @relation("ComponentUpdater")
}

//
model DataDictionary {
  id           String   @id @default(cuid())
  name         String   @db.VarChar(100)
  abbreviation String?  @db.VarChar(20)
  fullName     String   @db.VarChar(200)
  dataType     String?  @db.VarChar(50)
  description  String?  @db.Text
  isActive     Boolean  @default(true)
  version      String   @default("1.0") @db.VarChar(10)
  remarks      String?  @db.Text

  createdBy String
  updatedBy String?
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  creator User  @relation("DictionaryCreator", fields: [createdBy], references: [id])
  updater User? @relation("DictionaryUpdater", fields: [updatedBy], references: [id])

  @@map("data_dictionary")
}

//
model KnowledgeBase {
  id          String   @id @default(cuid())
  category    String   @db.VarChar(50)
  code        String   @unique @db.VarChar(20)
  title       String   @db.VarChar(200)
  description String   @db.Text
  solution    String?  @db.Text
  keywords    String?  @db.VarChar(500)
  attachments String?  @db.Text
  priority    String   @default("    ") @db.VarChar(10)
  status      String   @default("       ") @db.VarChar(20)

  createdBy String
  updatedBy String?
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  creator User  @relation("KnowledgeCreator", fields: [createdBy], references: [id])
  updater User? @relation("KnowledgeUpdater", fields: [updatedBy], references: [id])

  @@map("knowledge_base")
}

//
model ApiSystem {
  id          String   @id @default(cuid())
  systemCode  String   @unique @db.VarChar(20)
  systemName  String   @db.VarChar(100)
  systemType  String?  @db.VarChar(100)
  description String?  @db.Text
  version     String?  @db.VarChar(20)
  status      String   @default("          ") @db.VarChar(20)

  createdBy String
  updatedBy String?
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  creator      User             @relation("SystemCreator", fields: [createdBy], references: [id])
  updater      User?            @relation("SystemUpdater", fields: [updatedBy], references: [id])
  categories   FunctionCategory[]
  apiComponents ApiComponent[]

  @@map("api_systems")
}

//
model FunctionCategory {
  id          String   @id @default(cuid())
  systemId    String
  categoryCode String   @db.VarChar(50)
  categoryName String   @db.VarChar(100)
  description  String?  @db.Text
  isActive    Boolean  @default(true)

  createdBy String
  updatedBy String?
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  system       ApiSystem     @relation(fields: [systemId], references: [id], onDelete: Cascade)
  creator      User          @relation("CategoryCreator", fields: [createdBy], references: [id])
  updater      User?         @relation("CategoryUpdater", fields: [updatedBy], references: [id])
  apiComponents ApiComponent[]

  @@map("function_categories")
}

// API
model ApiComponent {
  id              String   @id @default(cuid())
  systemId        String
  categoryId      String
  name            String   @db.VarChar(100)
  version         String   @db.VarChar(20)
  description     String?  @db.Text
  developer       String?  @db.VarChar(100)
  endpointPath     String?  @db.VarChar(500)
  parameters      String?  @db.Text
  returnDescription String?  @db.Text
  dependencies     String?  @db.Text
  isActive         Boolean  @default(true)

  createdBy String
  updatedBy String?
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  system   ApiSystem       @relation(fields: [systemId], references: [id], onDelete: Cascade)
  category FunctionCategory @relation(fields: [categoryId], references: [id], onDelete: Cascade)
  creator  User            @relation("ComponentCreator", fields: [createdBy], references: [id])
  updater  User?           @relation("ComponentUpdater", fields: [updatedBy], references: [id])

  @@map("api_components")
}
```

## 5.

```bash
#        Prisma
npx prisma generate

#
npx prisma db push
```

```
✅ Your database is now in sync with your Prisma schema.
```

🎨

## 1.        Prisma

1. `src/lib`
2. `prisma.ts`

```typescript
```

```
import { PrismaClient } from '@prisma/client'

const globalForPrisma = globalThis as unknown as {
  prisma: PrismaClient | undefined
}

export const prisma = globalForPrisma.prisma ?? new PrismaClient()

if (process.env.NODE_ENV !== 'production') globalForPrisma.prisma = prisma
```

## 2.

src/app/page.tsx

typescript

```
'use client'

import { useState } from 'react'

export default function Home() {
 const [activeTab, setActiveTab] = useState('dictionary')

 //
 const dictionaryData = [
  { id: 1, name: 'MO', fullName: 'Manufacturing Order', type: 'VARCHAR(20)', description: '            ' },
  { id: 2, name: 'CUST_CODE', fullName: 'Customer Code', type: 'VARCHAR(10)', description: '         ' }
 ]

 const knowledgeData = [
  { id: 1, code: 'DEV001', title: 'React          ', category: '      ', status: '      ' },
  { id: 2, code: 'OPS001', title: '              ', category: '      ', status: '      ' }
 ]

 return (
  <div className="min-h-screen bg-gray-50">
   {/*          */}
   <header className="bg-white shadow-sm border-b">
    <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
     <div className="flex justify-between items-center py-4">
      <h1 className="text-2xl font-bold text-gray-900">           </h1>
      <button className="bg-blue-600 text-white px-4 py-2 rounded-md hover:bg-blue-700">

      </button>
     </div>
    </div>
   </header>

   <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-8">
    {/*          */}
    <div className="mb-8">
     <nav className="flex space-x-8">
      <button
       onClick={() => setActiveTab('dictionary')}
       className={`py-2 px-1 border-b-2 font-medium text-sm ${
        activeTab === 'dictionary'
         ? 'border-blue-500 text-blue-600'
         : 'border-transparent text-gray-500 hover:text-gray-700 hover:border-gray-300'
       }`}
    >

    </button>
    <button
      onClick={() => setActiveTab('knowledge')}
      className={`py-2 px-1 border-b-2 font-medium text-sm ${
        activeTab === 'knowledge'
          ? 'border-blue-500 text-blue-600'
          : 'border-transparent text-gray-500 hover:text-gray-700 hover:border-gray-300'
      }`}
    >

    </button>
    <button
      onClick={() => setActiveTab('systems')}
      className={`py-2 px-1 border-b-2 font-medium text-sm ${
        activeTab === 'systems'
          ? 'border-blue-500 text-blue-600'
          : 'border-transparent text-gray-500 hover:text-gray-700 hover:border-gray-300'
      }`}
    >

    </button>
  </nav>
</div>

{/*              */}
<div className="bg-white shadow-sm rounded-lg">
  <div className="px-6 py-4 border-b border-gray-200">
    <div className="flex justify-between items-center">
      <h2 className="text-lg font-medium text-gray-900">
        {activeTab === 'dictionary' && '                  '}
        {activeTab === 'knowledge' && '              '}
        {activeTab === 'systems' && '           '}
      </h2>
      <button className="bg-blue-600 text-white px-4 py-2 rounded-md text-sm hover:bg-blue-700">
        +
      </button>
    </div>
  </div>

  <div className="p-6">
    {/*                   */}
    {activeTab === 'dictionary' && (
  <div className="overflow-x-auto">
    <table className="min-w-full divide-y divide-gray-200">
      <thead className="bg-gray-50">
        <tr>
          <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
          <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
          <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
          <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
          <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
        </tr>
      </thead>
      <tbody className="bg-white divide-y divide-gray-200">
        {dictionaryData.map((item) => (
          <tr key={item.id} className="hover:bg-gray-50">
            <td className="px-6 py-4 whitespace-nowrap text-sm font-medium text-gray-900">{item.name}
            <td className="px-6 py-4 whitespace-nowrap text-sm text-gray-500">{item.fullName}</td>
            <td className="px-6 py-4 whitespace-nowrap text-sm text-gray-500">{item.type}</td>
            <td className="px-6 py-4 text-sm text-gray-500">{item.description}</td>
            <td className="px-6 py-4 whitespace-nowrap text-sm text-gray-500">
              <button className="text-blue-600 hover:text-blue-900 mr-4">      </button>
              <button className="text-red-600 hover:text-red-900">      </button>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
)}

{/*                    */}
{activeTab === 'knowledge' && (
  <div className="overflow-x-auto">
    <table className="min-w-full divide-y divide-gray-200">
      <thead className="bg-gray-50">
        <tr>
          <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
          <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
          <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
          <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
          <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
        </tr>
      </thead>
      <tbody className="bg-white divide-y divide-gray-200">
        {knowledgeData.map((item) => (
           <tr key={item.id} className="hover:bg-gray-50">
            <td className="px-6 py-4 whitespace-nowrap text-sm font-medium text-gray-900">{item.code}<
            <td className="px-6 py-4 whitespace-nowrap text-sm text-gray-500">{item.title}</td>
            <td className="px-6 py-4 whitespace-nowrap text-sm text-gray-500">{item.category}</td>
            <td className="px-6 py-4 whitespace-nowrap text-sm text-gray-500">
             <span className="inline-flex px-2 py-1 text-xs font-semibold rounded-full bg-green-100 text-gre
              {item.status}
             </span>
            </td>
            <td className="px-6 py-4 whitespace-nowrap text-sm text-gray-500">
             <button className="text-blue-600 hover:text-blue-900 mr-4">    </button>
             <button className="text-red-600 hover:text-red-900">    </button>
            </td>
           </tr>
          ))}
         </tbody>
        </table>
       </div>
      )}

      {/*         */}
      {activeTab === 'systems' && (
       <div className="text-center py-12">
        <p className="text-gray-500">                 ...</p>
       </div>
      )}
     </div>
    </div>
   </div>
  </div>
 )
}
```

🚀

## 1.

```
bash

npm run dev
```

## 2.

1. "Ready in XXXs"
2. http://localhost:3000
3.

## 3.

```bash
npx prisma studio
```

http://localhost:5555

✅

- ✅
- ✅
- ✅
- ✅

🆘

### 1 npm

Node.js "Add to PATH"

### 2 PostgreSQL

1. `.env.local`
2. PostgreSQL
3. "Services" postgresql

### 3 Prisma

1.
2. DATABASE_URL
3. `npx prisma db push`

### 4

1. `Ctrl + C`
2. `npm run dev -- -p 3001`

---

🎉

- ✅
- ✅
- ✅

1.
2.
3.
4.