# InkSpire: For Inspiring Writings and Ideas - Developer Documentation

Developed by **Saatvik Aggarwal** | Date: **05 October 2025**

## Table of Contents

1. [Project Overview](https://www.google.com/search?q=%231-project-overview)
2. [Installation & Setup](https://www.google.com/search?q=%232-installation--setup)
3. [System Architecture](https://www.google.com/search?q=%233-system-architecture)
4. [Database Design (PostgreSQL / Prisma)](https://www.google.com/search?q=%234-database-design-postgresql--prisma)
5. [API Documentation](https://www.google.com/search?q=%235-api-documentation)
6. [Frontend Functionality](https://www.google.com/search?q=%236-frontend-functionality)
7. [Deployment Guide](https://www.google.com/search?q=%237-deployment-guide)
8. [Future Enhancements](https://www.google.com/search?q=%238-future-enhancements)

## 1. Project Overview

**InkSpire** is a web-based platform designed to allow users to share, read, and engage with inspiring writings and ideas. It targets writers, readers, and content enthusiasts.

### Key Features

* User registration, login, and **authentication**
* Create, read, and comment on posts
* User profiles with personal posts and information
* Responsive frontend interface using **React.js**

### Tech Stack

| **Component** | **Technology** | **Description** |
| --- | --- | --- |
| **Frontend** | React.js, JavaScript, CSS, HTML | Client application |
| **Backend** | Node.js, Express.js | RESTful API Server |
| **Database** | **PostgreSQL** | Relational Database System |
| **ORM** | **Prisma** | Database toolkit and ORM |
| **Authentication** | JWT (JSON Web Tokens) | Secure user session management |
| **HTTP Client** | Axios | For client-side communication |

## 2. Installation & Setup

### Prerequisites

Ensure the following are installed and running on your system:

* **Node.js** and **npm**
* **PostgreSQL** server
* **Git**

### Setup Steps

1. **Clone the Repository**:  
   git clone [https://github.com/Saatvikaggarwal/InkSpire--for-inspiring-writings-and-ideas.git](https://github.com/Saatvikaggarwal/InkSpire--for-inspiring-writings-and-ideas.git)
2. **Install Dependencies**:  
   cd Backend && npm install  
   cd ../Client && npm install
3. **Configure Environment Variables (.env)**:
   * DATABASE\_URL: PostgreSQL connection string
   * JWT\_SECRET: Secret key for JWT
   * PORT: Server port
4. **Run Prisma Migrations**: This sets up the database schema.  
   npx prisma migrate dev  
   npx prisma generate
5. **Start the Backend Server**:  
   npm start
6. **Start the Frontend**:  
   npm start

## 3. System Architecture

### Module / Component Breakdown

#### Frontend (Client)

* **Pages**: Home, Post Detail, User Profile, Login, Register
* **Components**: Navbar, PostCard, PostForm, CommentSection, UserProfile
* **State Management**: React hooks (useState, useContext)
* **API Requests**: Axios with JWT headers

#### Backend (Server)

* **Core**: server.js/app.js (Server setup)
* **Logic**: routes/, controllers/, middleware/, helpers/
* **Database**: prisma/ (Prisma client integration)

### Main Application Flows (Sequence)

| **Flow** | **Steps** |
| --- | --- |
| **User Registration** | 1. Frontend submits form POST /api/auth/register 2. Backend validates, hashes password, stores in **PostgreSQL via Prisma** success response 3. Frontend redirects to login |
| **Create Post** | 1. Frontend submits new post POST /api/posts with JWT 2. Backend verifies JWT, creates post via **Prisma** returns post data 3. Frontend updates feed |

### Frontend-Backend Communication

* **Protocol**: RESTful API
* **Data Format**: JSON request/response
* **Headers**: Authorization (JWT), Content-Type

### Middleware Flow

* **Authentication**: Verifies JWT, attaches userId, blocks unauthorized requests.
* **Validation**: Checks required fields and returns errors if invalid.

## 4. Database Design (PostgreSQL / Prisma)

### Prisma Models / Tables

All primary and foreign keys use the UUID data type.

| **Table Name** | **Key Fields** | **Data Type** | **Relationships** |
| --- | --- | --- | --- |
| **Users** | id (UUID PK), username, email, passwordHash, createdAt | String / DateTime | **1:N** with Posts and Comments |
| **Posts** | id (UUID PK), userId (UUID FK Users), title, content, createdAt | String / DateTime | **1:N** with Comments |
| **Comments** | id (UUID PK), postId (UUID FK Posts), userId (UUID FK Users), content, createdAt | String / DateTime | N/A |

### Example Documents

**User**

{  
 "id": "uuid-user",  
 "username": "saatvik",  
 "email": "saatvik@example.com",  
 "passwordHash": "hashed\_password",  
 "createdAt": "2025-10-05T13:21:00Z"  
}

**Post**

{  
 "id": "uuid-post",  
 "userId": "uuid-user",  
 "title": "My First Post",  
 "content": "Hello world!",  
 "createdAt": "2025-10-05T13:25:00Z"  
}

## 5. API Documentation

### Register User

| **Detail** | **Value** |
| --- | --- |
| **Route** | POST /api/auth/register |
| **Authentication** | No |

**Request Body**

{  
 "username": "saatvik",  
 "email": "saatvik@example.com",  
 "password": "password123"  
}

**Response (Success)**

{  
 "success": true,  
 "message": "User registered successfully"  
}

**Sample cURL**

curl -X POST http://localhost:5000/api/auth/register -H "Content-Type: application/json" -d '{"username":"saatvik","email":"saatvik@example.com","password":"password123"}'

### Create Post

| **Detail** | **Value** |
| --- | --- |
| **Route** | POST /api/posts |
| **Authentication** | Yes (JWT) |

**Request Body**

{  
 "title": "My First Post",  
 "content": "Hello world!"  
}

**Response (Success)**

{  
 "success": true,  
 "postId": "uuid-post"  
}

## 6. Frontend Functionality

### Page Breakdown

| **Page** | **Components** | **Key Functionality** |
| --- | --- | --- |
| **Home** | PostCard | Main post feed. |
| **Post Detail** | CommentSection | View single post and comments. |
| **User Profile** | N/A | Display user information and their posts. |
| **Login/Register** | Authentication forms | Handle user sign-in/sign-up. |

### User Interaction Flow

1. **Login** JWT stored in context/state.
2. **Dashboard** fetches posts.
3. **Create post** POST /api/posts update feed.
4. **Add comment** POST /api/comments update comments dynamically.

## 7. Deployment Guide

### Hosting Options

| **Component** | **Options** |
| --- | --- |
| **Backend** | Heroku, AWS EC2, Vercel |
| **Frontend** | Vercel, Netlify, S3 |

### Production Configuration

* Set **Environment variables**: DATABASE\_URL, JWT\_SECRET, PORT.
* Enable **CORS** for the frontend domain.

### Build Commands

# Backend  
cd Backend  
npm install  
npm start  
  
# Frontend (Build)  
cd Client  
npm install  
npm run build  
serve -s build

## 8. Future Enhancements

* Real-time notifications for posts/comments
* Search and filter posts
* Social login (Google/Facebook)
* Enhanced responsive UI with animations