Viewstats Interview Documentation

Documentation by **Jean Pierre Barnett**

1. Introduction

Welcome to the Viewstats Full Stack Project, developed by Jean Pierre Barnett. This project represents a comprehensive web application that allows users to register and log in to manage their to-do lists.

A-) Features

The application offers the following features:

- User registration and login.
- Creation, management, and deletion of to-do list items.
- Access to to-do lists from anywhere using the user account.

B-) Technologies Used

The project is developed using the following technologies:

- Frontend: Next.js and React.
- Backend: Node.js, Express, and MongoDB.

C-) UI/UX Design

Special attention has been given to user interface design to provide an intuitive and efficient experience to users. Features include:

- Loading and error states.
- Custom and thoughtful user interface.
- Mobile-friendly and accessibility.
- Animations and tooltips.

Viewstats

Interview

Documentation

Documentation by **Jean Pierre Barnett**

2. Expected Outcomes

The project is expected to meet the following criteria:

- Flawless setup of the Next.js application with advanced configurations.
- Fully responsive design with exceptional UI/UX.
- Excellent use of state management using React state/Context API.
- Seamless integration of API with error handling.
- Well-designed backend models and architecture.
- Comprehensive and clear documentation.
- Quality and efficiency of the code.

The project should be submitted via the Vercel platform, providing the deployment URL on Vercel as well as access to the backend repository.

2. Help and technical support

If you have any questions, please use Contact Us via WhatsApp for help:

WHATSAPP SUPPORT

3. Environment Setup

To set up the Next.js application with MongoDB Atlas and the Next.js server, follow these steps:

A. Install Node.js

TOP

I. Download instructions:
   - Mac users: For OS X, download the .pkg file from the following link. **(v20.5.0 recommended)**
   - Windows users: Download the MSI or .exe form below, whichever you prefer. **(v20.5.0 recommended)**

DOWNLOAD NODE.JS V.20.5.0 FROM HERE

II. After installing Node.js, you may need to restart your computer.
To ensure Node.js is installed correctly, you can type the following in the Windows console:

```
node -v
```

NOTE: DO NOT IGNORE THE NODE JS VERSION. IT MUST BE 20.5.x

B. Install Git and Windows-Build-Tools (Windows PC Only)

- Download and install the latest version from this link: https://git-scm.com/

- Run PowerShell as administrator and execute
```
npm install -g windows-build-tools
```

C. Install Yarn

In your command prompt, type:

```
npm install -g yarn
```

D. Set Up Environment Variables and Configuration

Create a .env file in the root directory of your Next.js application and configure the following variables:

```
MONGODB_URI=
```

TOP

Replace with your actual MongoDB Atlas connection string.

For the Next.js server configuration, refer to the official Next.js documentation: https://nextjs.org/docs/api-reference/cli#production

Now you're ready to start developing your Next.js application with MongoDB Atlas!

4. Installing the Next.js Application

Step 1: Create a new Next.js project

```
npx create-next-app your_project_name cd
your_project_name
```

Step 2: Install additional dependencies if needed

```
npm install
```

Step 3: Configure and define application routes

```
// pages/index.js
// pages/about.js
// Other page and route files
```

Step 4: Set up backend if needed

```
src\lib\utils.js import mongoose from "mongoose"
const connection = {}; export const connectToDb =
async () => { try { if(connection.isConnected) {
console.log("Using existing connection"); return;
} const db = await
mongoose.connect(process.env.MONGO);
connection.isConnected =
db.connections[0].readyState; } catch (error) {
console.log(error); throw new Error(error); } };
```

TOP

Step 5: Run the development server

```
npm run dev
```

Viewstats Interview Documentation

Documentation by **Jean Pierre Barnett**

You have now successfully set up a Next.js application. You can further customize and expand your application using Next.js and other related technologies.

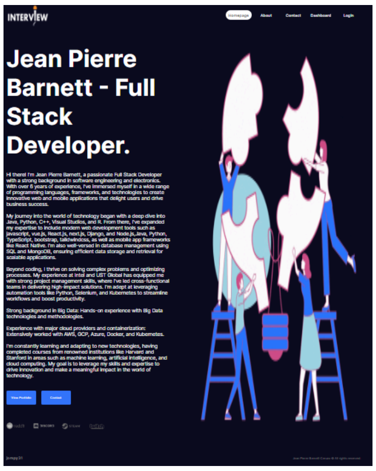
4. Web Application Deployment

Overview of sections of the application:

When you run the deployed application, you will find the following sections:

1. Homepage /

The homepage of ViewStats showcases Jean Pierre Barnett's profile as a Full Stack Developer, emphasizing his skills and experience. It includes a brief summary of his expertise and an invitation to view his portfolio and contact him.



Jean Pierre Barnett - Full Stack Developer.

Hi there! I'm Jean Pierre Barnett, a passionate Full Stack Developer with a strong background in software engineering and electronics. With over 6 years of experience, I've immersed myself in a wide range of programming languages, frameworks, and technologies to create innovative web and mobile applications that delight users and drive business success.

My journey into the world of technology began with a deep dive into Java, Python, C++, Visual Studios, and R. From there, I've expanded my expertise to include modern web development tools such as javascript, vue.js, React.js, next.js, Django, and Node.js,Java, Python, TypeScript, bootstrap, taildwindcss, as well as mobile app frameworks like React Native. I'm also well-versed in database management using SQL and MongoDB, ensuring efficient data storage and retrieval for scalable applications.

Beyond coding, I thrive on solving complex problems and optimizing processes. My experience at Intel and UST Global has equipped me with strong project management skills, where I've led cross-functional teams in delivering high-impact solutions. I'm adept at leveraging automation tools like Python, Selenium, and Kubernetes to streamline workflows and boost productivity.

Strong background in Big Data: Hands-on experience with Big Data technologies and methodologies.

Experience with major cloud providers and containerization: Extensively worked with AWS, GCP, Azure, Docker, and Kubernetes.

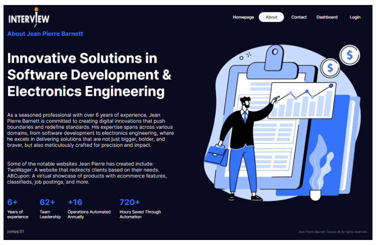
I'm constantly learning and adapting to new technologies, having completed courses from renowned institutions like Harvard and Stanford in areas such as machine learning, artificial intelligence, and cloud computing. My goal is to leverage my skills and expertise to drive innovation and make a meaningful impact in the world of technology.

```
The page.js document shows a React component
called Home, which represents the home page of a
website. In this component, elements like Link and
Image are imported from the next/link and
next/image library respectively. The Home
component includes a main container with two
sections: one for the textual content and another
```

> for the image. The textual content describes Jean Pierre Barnett's experience and skills as a Full Stack developer, highlighting his experience in a variety of technologies and tools. Additionally, links are provided to view his portfolio and to contact him. The component uses CSS styles defined in the home.module.css file to format and style elements on the page.

2. About

The src\app\about\page.jsx file displays the code for an About page of a website. Imports the necessary Next.js modules, including Image, and page-specific styles from a about.module.css file. The AboutPage function defines the content of the About page, which includes a subtitle, a title, a detailed description about Jean Pierre Barnett and his professional achievements, as well as a list of websites notably created by him. Additionally, it displays key statistics about his experience and achievements, represented in the form of boxes. The page is designed to highlight Jean Pierre Barnett's experience and skills in software development and electronic engineering.



About Jean Pierre Barnett

Innovative Solutions in Software Development & Electronics Engineering

As a seasoned professional with over 6 years of experience, Jean Pierre Barnett is committed to creating digital innovations that push boundaries and redefine standards. His expertise spans across various domains, from software development to electronics engineering, where he excels in delivering solutions that are not just bigger, bolder, and braver, but also meticulously crafted for precision and impact.

Some of the notable websites Jean Pierre has created include:
TwoWager: A website that redirects clients based on their needs.
ABCupon: A virtual showcase of products with ecommerce features, classifieds, job postings, and more.

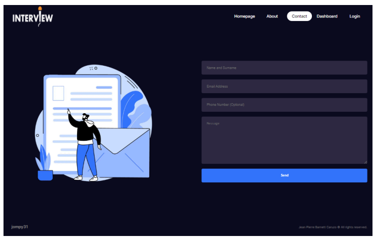
6+ Years of experience
62+ Team Leadership
+16 Operations Automated Annually
720+ Hours Saved Through Automation

jompy31

Jean Pierre Barnett Caruzo © All rights reserved.

3. Contact

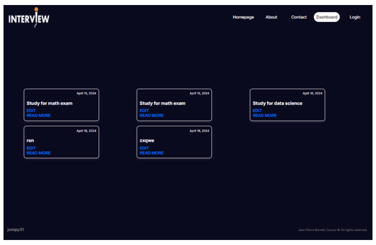
The contact.js file displays the code for a Contact page on a website. Import the necessary React modules, such as useRef, and the necessary Next.js image. It also imports page-specific styles from a contact.module.css file. The ContactPage feature defines the content of the contact page, which includes a contact form where users can enter their name, email, phone number (optional), and a message. When the user submits the form, the form data is collected and a WhatsApp message is generated with this information. Finally, a new browser window opens with the WhatsApp link containing the pre-filled message. The page is designed to allow users to contact Jean Pierre Barnett via WhatsApp.



4. Dashboard

The page.jsx file in the src\app\dashboard folder displays the blog page code. This page renders a list of blog posts obtained from a local API. Let's start with the Dashboard function, which uses the useState hook to maintain a local state of posts and the useEffect hook to make a data request to the server when the component is first mounted. This request is made through the fetchData function, which fetches data from the local API /api/blog and updates the posts status with the response. The Dashboard function then renders the posts inside a div container with the container class. Iterates over the list of posts using the map method, and for each post, renders a PostCard component, passing the post as a prop. The PostCard component, located in src\components\postCard\postCard.jsx, receives the post as a prop and displays information such as the post image (if available), the post date, the title, and a short description of the post . Additionally, it provides links to edit the post if you are authenticated or just read more about it. Finally, in src\app\dashboard\[slug]\page.jsx, we find the code for an individual post page. This page receives a slug as a URL parameter and uses this slug to get the post details from the local API. Renders the post image (if available), title, author information, and post date, as well as the full content of the post. Both pages make use of functional React components, hooks, and make data requests to a local API to dynamically display blog information.

TOP



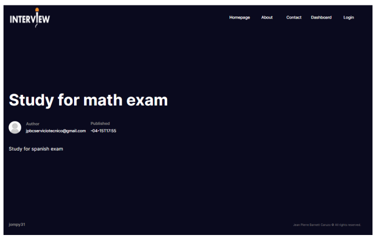
April 15, 2024 Study for math exam

April 15, 2024 Study for math exam

April 16, 2024 Study for data science

April 16, 2024 ren

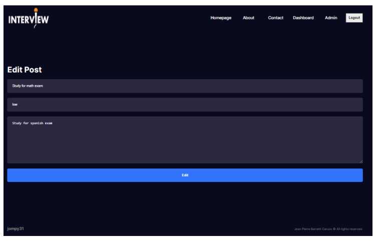
April 16, 2024 zxqwe



Study for math exam

Author jpbcserviciotecnico@gmail.com Published -04-15T17:55

Study for spanish exam



Edit Post

Study for math exam

low

Study for spanish exam

5. Register

Register Page (page.jsx):

In RegisterPage, the RegisterForm component is utilized to display the registration form. This component resides within a container styled with the

```
styles.container
```

class defined in the

```
register.module.css
```

style file.

Register Form Component (registerForm.jsx):

This component encompasses a form including fields for username, email, password, and password confirmation. Upon form submission, the entered data is dispatched to the

```
register
```

function defined in the

```
action.js
```

file to process user registration.

The

```
register
```

function conducts several checks before registering the user into the database. It ensures password match, verifies unique username, and validates the entered password against the one stored in the database if the user exists. Upon successful registration, the user is redirected to the login page. In case of any error, an appropriate error message is displayed.

Additionally, the form includes a link redirecting users to the login page if they already have an account.

Registration Actions (action.js):

The

```
action.js
```

file contains the

```
register
```

function responsible for registering a new user in the database. This function verifies the validity of the user's input data, such as ensuring a unique username and matching passwords. It then uses bcrypt to hash the password before saving it to the database. If the registration is successful, it returns an object with a success indicator. If there's an error during registration, it returns an object with an appropriate error message.

The file also includes other related actions such as

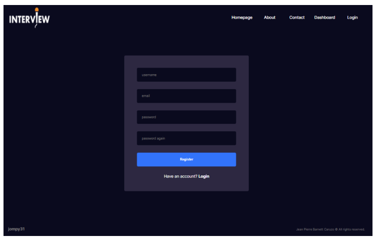
```
login
```

and

```
logout
```

functions for handling user authentication.

TOP



6. login

Login Page (page.jsx):

In the LoginPage component located in src\app\auth\login\page.jsx, the LoginForm component is utilized to display the login form. This component resides within a container styled with the

```
styles.container
```

class defined in the

```
login.module.css
```

style file.

Login Form Component (loginForm.jsx):

The LoginForm component displays a form with fields for username and password. Upon form submission, the entered data is dispatched to the

```
login
```

function defined in the

```
action.js
```

file to authenticate the user.

The

```
login
```

function verifies the user's credentials against the store data in the database. If the credentials are correct, the

TOP

user is authenticated, and if not, an appropriate error message is displayed.

If the user doesn't have an account, there is a link provided to the registration page.

Additionally, the LoginForm component includes details for a demo admin user for testing purposes.

Authentication Actions (action.js):

The

```
action.js
```

file contains the

```
login
```

function responsible for authenticating the user. This function checks whether the provided username and password match an existing user in the database. If the authentication is successful, it returns an object with a success indicator. If there's an error during authentication, it returns an object with an appropriate error message.

Authentication Configuration (auth.config.js, auth.js):

Similar to the registration process, authentication configuration and logic are handled in the

```
auth.config.js
```

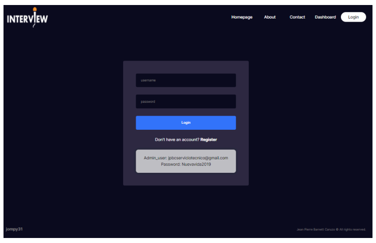
and

```
auth.js
```

files. These files define the authentication pages, providers, and handle authentication callbacks.

TOP

The user's authentication status is maintained by NextAuth.js. Upon successful login, NextAuth.js generates a JWT (JSON Web Token) containing the user's information, which is stored in a session cookie. This allows the user to remain authenticated until they log out or the session expires.


INTERVIEW
Homepage
About
Contact
Dashboard
Login
username
password
Login
Don't have an account? **Register**

Admin_user: jpbcserviciotecnico@gmail.com
Password: Nuevavida2019

jompy31
Jean Pierre Barnett Caruzo © All rights reserved.

7. Admin

Admin Dashboard Page (page.jsx):

The AdminPage component located in src\app\admin\page.jsx serves as the dashboard for managing posts and users. It consists of two main sections: one for managing posts and another for managing users. Each section displays relevant components and forms for adding, editing, and deleting posts or users.

The session information is obtained using the

```
auth()
```

function, which authenticates the user. The user's ID is then passed as a prop to the AdminPostForm component.

Admin Posts Component (adminPosts.jsx):

The AdminPosts component displays a list of all posts fetched from the database using the

```
getPosts()
```

function. Each post is rendered with its title, an optional image, and options to edit or delete the post.

The edit option redirects to the admin edit form page, passing the post ID as a URL parameter.

Admin Post Form Component (adminPostForm.jsx):

The AdminPostForm component provides a form for creating new posts. It includes fields for entering the title, description, and user ID, with the user ID being pre-filled using the session information obtained from authentication.

Upon form submission, the data is dispatched to the

```
addPost
```

function defined in the

```
action.js
```

file for adding a new post to the database.

Admin Users Component (adminUsers.jsx):

The AdminUsers component displays a list of all users fetched from the database using the

```
getUsers()
```

function. Each user is rendered with their username, an optional avatar image, and an option to delete the user.

Admin User Form Component (adminUserForm.jsx):

The AdminUserForm component provides a form for adding new users. It includes fields for entering the username, email, password, and specifying whether the user is an admin or not.

Upon form submission, the data is dispatched to the

```
addUser
```

function defined in the

```
action.js
```

file for adding a new user to the database.

Admin Edit Form Page (adminEditForm.jsx):

The AdminEditForm component serves as the page for editing existing posts. It retrieves the post ID from the URL parameters and fetches the corresponding post data from the database. The retrieved data is then pre-filled into the form fields, allowing the user to make edits.

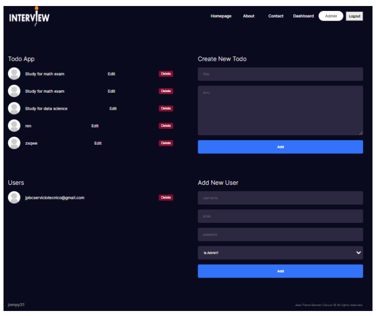
Upon form submission, the edited data is dispatched to the

```
editPost
```

function defined in the

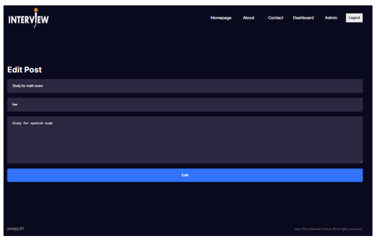
```
action.js
```

file for updating the post in the database.



Study for math exam Edit Delete
Study for math exam Edit Delete
Study for data science Edit Delete
ren Edit Delete
zxqwe Edit Delete

jpbcserviciotecnico@gmail.com Delete



Study for math exam
low
Study for spanish exam

6. User Roles and Functions

When accessing the application via a web browser, different user roles have distinct functions and capabilities. Below is a breakdown of user roles and their respective functions:

6.1. User

The User role represents the target audience for the products and services provided by the application. Upon accessing the application, the User can:

- Register an account
- Login to the application
- Edit tasks

6.2. Admin User

This user role is aimed at:

- Creating tasks
- Editing tasks
- Deleting tasks
- Creating users
- Granting admin privileges to users

If you have any questions, please use Contact Us via WhatsApp for help:

WHATSAPP SUPPORT

TOP