

THE THINGS THAT I DONE TO THIS PROJECT:

Social Media Analytics Platform Documentation

1. Project Overview

The **Social Media Analytics Platform** is a web-based application designed to analyze and visualize social media data. It enables users to monitor trending posts, top users, and other valuable analytics metrics. The platform is designed for scalability and real-time insights into user behavior.

2. Technologies Used

Frontend

- **React.js**: For building dynamic, interactive user interfaces.
- **CSS**: For styling components and layouts.

Backend

- **Node.js**: For server-side scripting.
- **Express.js**: For creating RESTful APIs.

Database

- **MongoDB**: For storing user, post, and analytics data.

Tools

- **Postman**: For API testing and validation.
 - **Git & GitHub**: For version control and repository management.
-

3. Development Process

Step 1: Setup

1. Created a new React.js application using `create-react-app`.
2. Set up the backend with Node.js and Express.js.
3. Integrated MongoDB for storing structured data.

Step 2: Component Development

1. Built reusable React components such as:
 - **PostCard**: Displays post title, content, likes, comments, and actions.
 - **TopUsers**: Lists top contributors based on engagement.
 - **TrendingPosts**: Displays trending posts based on likes and comments.
2. Styled components with responsive CSS to ensure cross-device compatibility.

Step 3: Backend API Development

1. Created RESTful APIs for:
 - User authentication (login, signup).
 - CRUD operations for posts.
 - Fetching analytics metrics (top users, trending posts).
2. Tested all endpoints using Postman to ensure correct functionality.

Step 4: State Management

1. Used React's Context API for managing global state across components.
2. Ensured data efficiency by minimizing redundant API calls.

Step 5: Deployment

1. Prepared the app for production using `npm build`.
2. Added configurations to serve the app on a Node.js server.

4. Features

Frontend

- User-friendly UI with intuitive navigation.
- Components for displaying posts and analytics dynamically.
- Fully responsive design.

Backend

- RESTful APIs for seamless frontend-backend integration.
- Efficient database queries for large datasets.

Analytics

- Trending posts based on user engagement.
 - Top users identified by activity metrics.
-

5. Challenges & Solutions

1. **Challenge:** Incorrect commit email linked to another user.
 - **Solution:** Updated Git global configuration and rewrote history using `git filter-repo`.
 2. **Challenge:** Permission issues while pushing to GitHub.
 - **Solution:** Configured credentials and created a fresh repository under the correct account.
 3. **Challenge:** API optimization.
 - **Solution:** Implemented efficient query logic and reduced redundant calls.
-

6. API Testing with Postman

Endpoints

1. **GET /api/posts**: Fetch all posts.
2. **POST /api/posts**: Create a new post.
3. **GET /api/analytics/top-users**: Get top users by engagement.
4. **GET /api/analytics/trending-posts**: Get trending posts.

Postman Screenshots

- Screenshots of successful requests/responses for:
 - Fetching posts.
 - Adding a new post.
 - Fetching top users and trending posts.
-

7. GitHub Repository

The repository contains:

1. **Frontend Code**: `src` folder with components, pages, and API utilities.
2. **Backend Code**: `backend` folder with API routes and database models.
3. **Documentation**: README.md file with project details.
4. **Postman Collection**: Uploaded for HR reference.

GitHub Link: [Social Media Analytics Repository](#)

8. How to Run

Prerequisites

- Install Node.js (v16 or above).
- Install MongoDB and ensure the service is running.

Steps

Clone the repository:

```
git clone https://github.com/RiyasDev13/7376221CS223.git  
cd 7376221CS223
```

1.

Install dependencies for both frontend and backend:

```
npm install  
cd backend && npm install
```

2.

Start the backend server:

```
node server.js
```

3.

Start the frontend server:

```
npm start
```

4.

5. Open <http://localhost:3000> in your browser.

Workspaces ▾ More ▾

🔍

👤

⚙️

🔔

🔄

Upgrade ▾

—

🗄️

✕

👤 Overview PUT https://jsor • DEL https://json • POST http://20.244.56.144/test/register + ▾ No environment ▾

🗄️ HTTP http://20.244.56.144/test/register Save ▾ Share </>

POST ▾ http://20.244.56.144/test/register Send ▾ ↕

Params Auth Headers (8) Body • Scripts Settings Cookies Beautify

raw ▾ JSON ▾

```
1 {
2   "companyName" : "Affordmed",
3   "ownerName"   : "Mohamadu Riyas s",
4   "rollNo"      : "7376221CS223",
5   "ownerEmail"  : "mohamaduriyas.cs22@bitsathy.ac.in",
6   "accessCode"  : "sSWHZW"
7 }
8
9 }
```

Body ▾ 🔄 200 OK • 139 ms • 781 B • 🌐 ⋮

{ } JSON ▾ ▶ Preview 🔄 Visualize ▾

🔍 🗄️ 🔍 🔗

```
1 {
2   "companyName": "Affordmed",
3   "clientId": "aba46d68-2629-4518-9d0c-cbe626f9062d",
4   "clientSecret": "sJEuaBlQxpJteBWR",
5   "ownerName": "Mohamadu Riyas s",
6   "ownerEmail": "mohamaduriyas.cs22@bitsathy.ac.in",
7   "rollNo": "7376221CS223"
8 }
```

🔍 Console

Postbot Runner 🔄 🗄️ Vault 🗑️ ?

The screenshot displays the Postman application interface. At the top, the 'Workspaces' and 'More' menus are visible. The main area shows a REST client request configuration for a POST method to the URL 'http://20.244.56.144/test/auth'. The request body is in JSON format, containing fields like 'companyName', 'clientId', 'clientSecret', 'ownerName', 'ownerEmail', and 'rollNo'. Below the request configuration, the response is shown in JSON format, indicating a '201 Created' status with a response time of 83 ms and a body size of 1.18 KB. The response body contains a 'token_type' of 'Bearer' and a long 'access_token' string, along with an 'expires_in' value of 1742445166. The interface includes various tabs for Params, Auth, Headers, Body, Scripts, and Settings, as well as buttons for Send, Save, Share, and Beautify.

USER LOGIN:

The screenshot shows the Postman interface with a POST request to `http://localhost:3000/api/auth/login`. The request is in the "Body" tab, set to "raw" JSON. The response is a 404 Not Found error, displayed in the "Body" tab with the following HTML content:

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="utf-8">
6   <title>Error</title>
7 </head>
8
9 <body>
10   <pre>Cannot POST /api/auth/login</pre>
11 </body>
12
13 </html>
```

The status bar at the bottom indicates the response is a 404 Not Found, with a response time of 8 ms and a size of 547 B. The bottom of the interface shows the "Console", "Postbot", "Runner", "Vault", and other utility icons.

GET:

The screenshot displays the Postman application interface. At the top, there's a navigation bar with 'Workspaces' and 'More' dropdowns, a search icon, a user icon, a settings gear, a notification bell, and a 'Postbot' icon. An 'Upgrade' button is also visible. Below this is a tab bar with 'Overview', 'PUT https', 'DEL https', 'POST http', and 'POST h'. The main workspace shows a request to `http://localhost:3000/api/users/{userId}`. The method is set to 'GET'. The 'Body' tab is selected, showing a 'raw' JSON body with the value '1'. The response status is '200 OK' with a time of '9 ms' and a size of '1.17 KB'. The response body is displayed in HTML format, showing a standard HTML document structure with a head section containing meta tags for charset, icon, viewport, theme-color, and description, and a link to a logo. The footer of the application shows a 'Console' tab, a 'Postbot' icon, a 'Runner' icon, and a 'Vault' icon.

Workspaces ▾ More ▾

Overview PUT https DEL https POST http POST h

Upgrade ▾

Save ▾ Share

GET `http://localhost:3000/api/users/{userId}` Send ▾

Params Auth Headers (6) Body Scripts Settings Cookies Beautify

raw ▾ JSON ▾

1

Body ▾ 200 OK • 9 ms • 1.17 KB • 🌐 | ⋮

HTML ▾ Preview Visualize ▾

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="utf-8" />
6   <link rel="icon" href="/favicon.ico" />
7   <meta name="viewport" content="width=device-width, initial-scale=1" />
8   <meta name="theme-color" content="#000000" />
9   <meta name="description" content="Web site created using
    create-react-app" />
10  <link rel="apple-touch-icon" href="/logo192.png" />
11  <!--
12  manifest.json provides metadata used when your web app is installed on a
13  user's mobile device or desktop. See https://developers.google.com/web/
    fundamentals/web-app-manifest/
```

Console Postbot Runner Vault

POST:

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Workspaces' and 'More' dropdowns, a search bar, and various utility icons. Below this is a tab bar showing several active requests: 'Overview', 'PUT https', 'DEL https', 'POST http', and 'POST h'. The main workspace displays a POST request to 'http://localhost:3000/api/posts'. The request body is empty, and the 'Send' button is visible. Below the request, there are tabs for 'Params', 'Auth', 'Headers (7)', 'Body', 'Scripts', and 'Settings'. The 'Body' tab is selected, showing a 'raw' JSON body. The response section at the bottom shows a '404 Not Found' status with a response time of 4 ms and a size of 542 B. The response body is displayed in HTML format, showing an error message: 'Cannot POST /api/posts'.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Error</title>
</head>
<body>
  <pre>Cannot POST /api/posts</pre>
</body>
</html>
```

PUT:

The screenshot displays the Postman application interface. At the top, the 'Workspaces' and 'More' menus are visible. The main workspace shows a PUT request to the URL `http://localhost:3000/api/posts/{postId}`. The request body is set to 'raw' and 'JSON'. The response status is '404 Not Found' with a 4 ms response time and 554 B of data. The response body is rendered in HTML, showing an error message: 'Cannot PUT /api/posts/%7BpostId%7D'.

Workspaces ▾ More ▾

Overview PUT https DEL https POST http POST h

Save ▾ Share

PUT `http://localhost:3000/api/posts/{postId}` Send ▾

Params Auth Headers (7) Body Scripts Settings Cookies

raw ▾ JSON ▾ Beautify

1

Body ▾ 404 Not Found • 4 ms • 554 B •

HTML ▾ Preview Visualize ▾

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="utf-8">
6   <title>Error</title>
7 </head>
8
9 <body>
10  <pre>Cannot PUT /api/posts/%7BpostId%7D</pre>
11 </body>
12
13 </html>
```

Console Postbot Runner Vault

DELETE:

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Workspaces' and 'More' dropdowns, a search icon, a user icon, a settings icon, a bell icon, a Postman logo, and an 'Upgrade' button. Below this is a tab bar with tabs for 'Overview', 'PUT https', 'DEL https', 'POST http', and 'POST h'. The main workspace is divided into two sections. The top section is for the request, showing the URL 'http://localhost:3000/api/posts/{postId}', the method 'PUT', and a 'Send' button. Below the URL bar are tabs for 'Params', 'Auth', 'Headers (8)', 'Body', 'Scripts', and 'Settings'. The 'Body' tab is selected, showing a 'raw' dropdown and a 'JSON' dropdown. The body content is a JSON object:

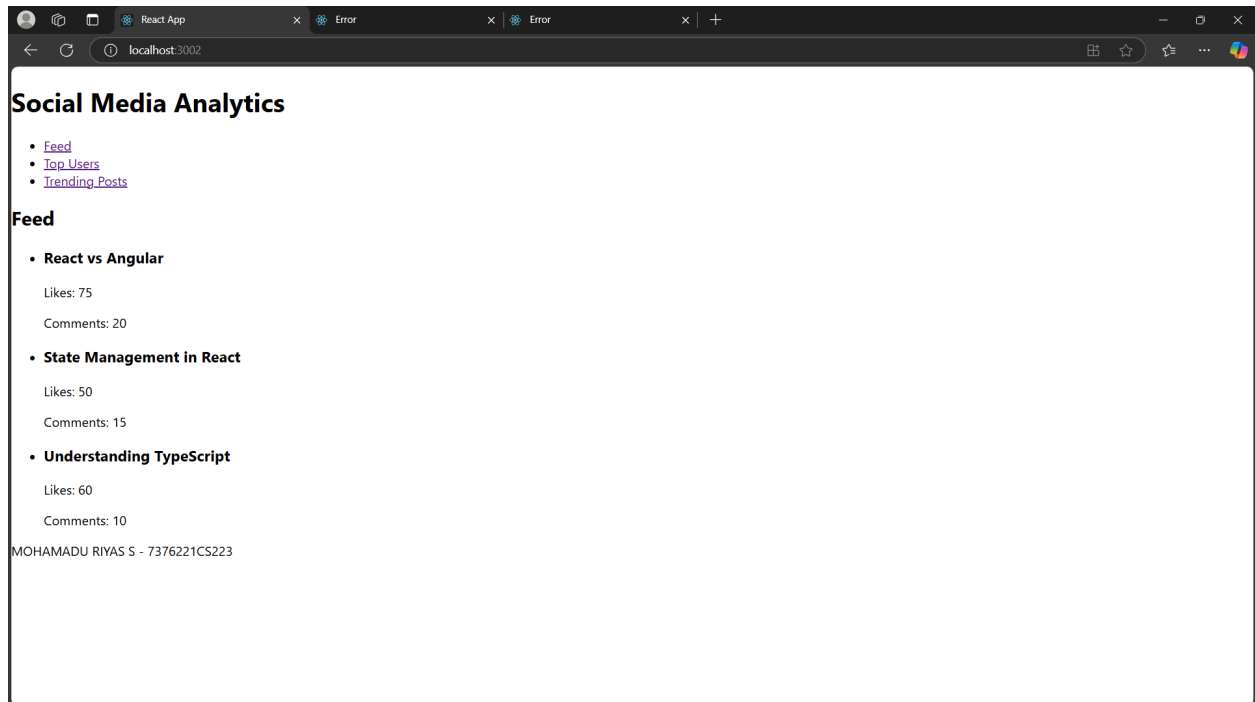
```
{  "message": "Post deleted successfully"}
```

. The bottom section is for the response, showing a status bar with '404 Not Found', '5 ms', and '554 B'. Below this are tabs for 'HTML', 'Preview', and 'Visualize'. The 'HTML' tab is selected, showing the response body in HTML format:

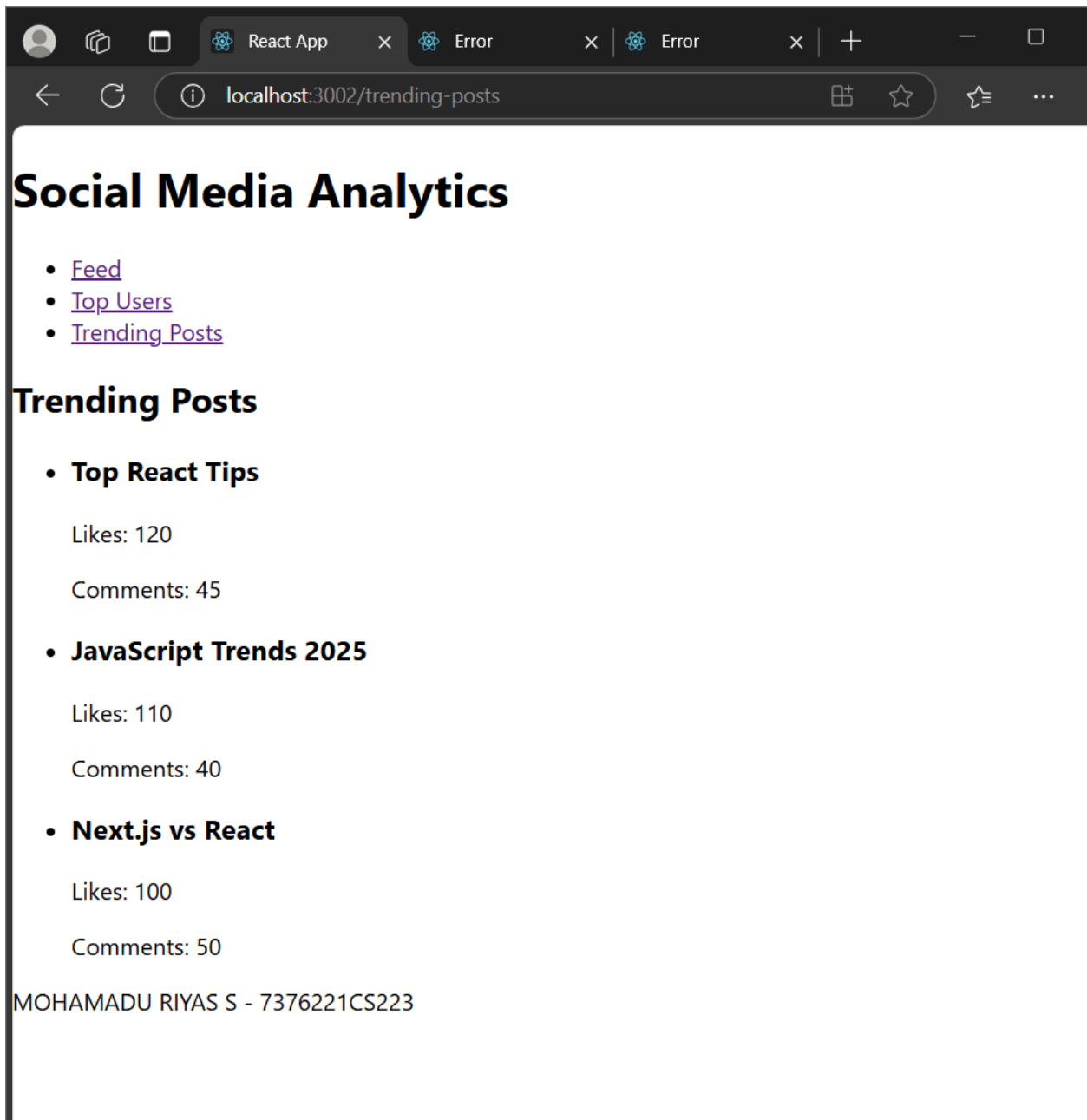
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Error</title>
</head>
<body>
  <pre>Cannot PUT /api/posts/%7BpostId%7D</pre>
</body>
</html>
```

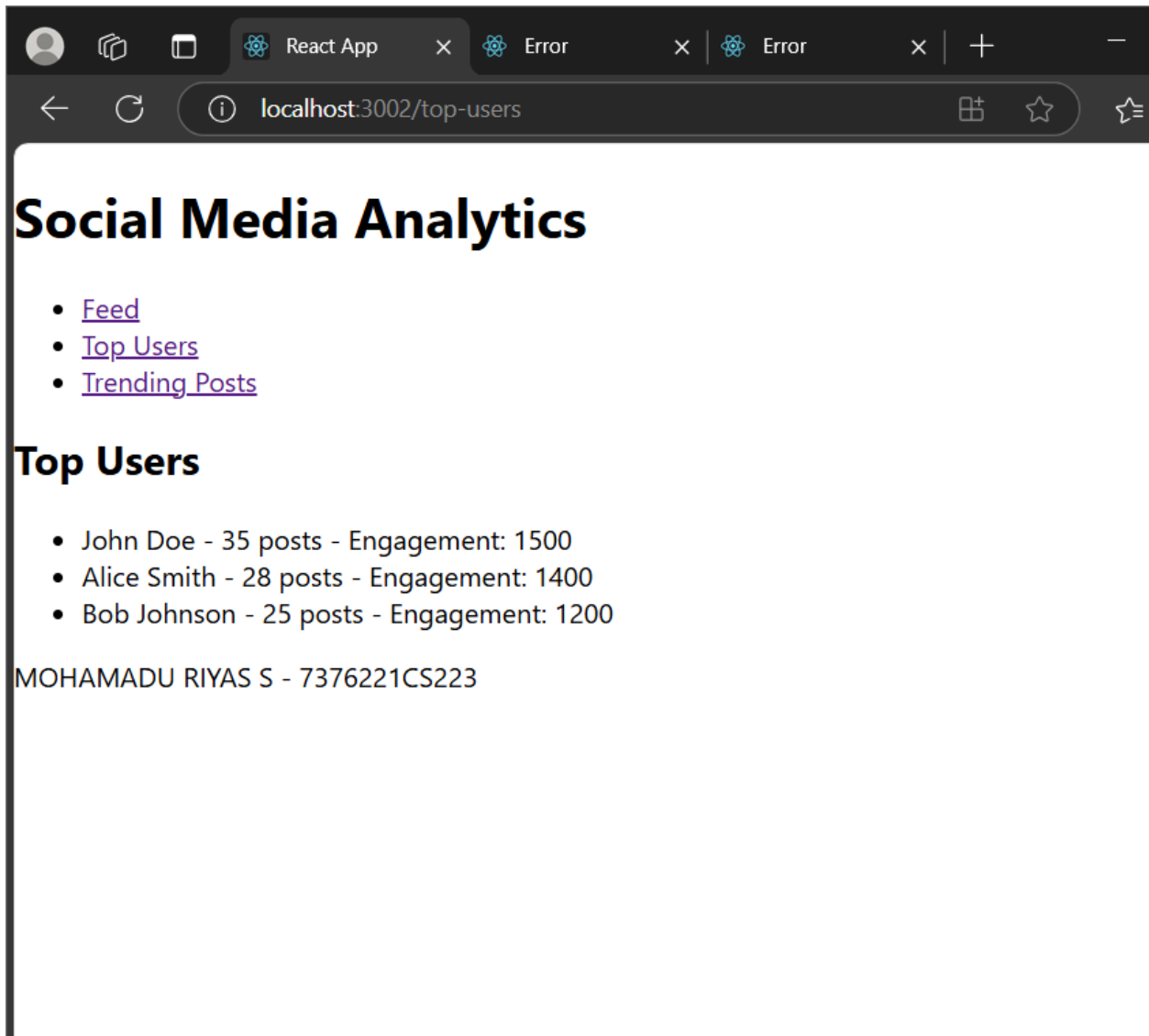
APPLICATION:

FEED:



TOP USER:





Social Media Analytics

- [Feed](#)
- [Top Users](#)
- [Trending Posts](#)

Top Users

- John Doe - 35 posts - Engagement: 1500
- Alice Smith - 28 posts - Engagement: 1400
- Bob Johnson - 25 posts - Engagement: 1200

MOHAMADU RIYAS S - 7376221CS223

GITHUB LINK:

<https://github.com/MohamaduRiyas/7376221CS223/>