International Journal of Trend in Scientific Research and Development (IJTSRD)
Special Issue on Smart Innovations in Computer Science and Applications
Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Event Driven Notification Broadcasting Service Web Development

Rufaiz Ansar Baig
Department of Master of Computer Application,
G. H. Raisoni Skill Tech University, Nagpur, India

Abstract
Web applications today need to talk to users away. This makes the application more interesting and responsive. Old systems that constantly check for updates have delays and waste resources.

This project is about a service that sends notifications when something happens in the system. It does this in a way.

From the perspective of the person building the user interface the system is about creating an interface that can change and respond quickly. It gets notifications. The system can handle users and is efficient. It is suitable for all kinds of applications, such, as shopping, healthcare and education.

Keywords: Event-Driven Architecture, Real-Time Notifications, Frontend Development, WebSockets, UI/UX, Broadcasting Service, JavaScript, Responsive Design

Introduction

We want our website to talk to people in a way. The old method of checking all the time makes our website slow. Uses a lot of server power. Our new system is different. It will tell people things soon as they happen. We want the part of our website that people see to be easy to use.

Our website should be able to change what is on the page without reloading. This way people will have a time on our website. Our website will update things so people can see what is new. Our website will be fast and fun. Our website will be an experience, for people.

Proposed AI-Based Solution

Although the system mainly works with events using AI ideas can make it better by:

- Predicting what users will do to send them notifications
- Making important alerts show up first
- Cutting down on notifications, with smart filters

The part of the system that users see can include showing notifications in order of importance and changing how the interface looks based on how users interact with it.

Motivation

People want to chat with each other when using modern apps. The old method of checking for updates has its downsides. Users want an experience when using an app. A responsive user interface is crucial when building web stuff. You need to handle notifications to many users quickly and smartly. Real-time communication is key for apps. They need to make sure users can talk to each

other in time. Modern apps rely on real-time communication to work well. It's essential, for an user experience.People want to chat with each other when using modern apps. The old method of checking for updates has its downsides. Users want an experience when using an app. A responsive user interface is crucial when building web stuff. You need to handle notifications to many users quickly and smartly. Real-time communication is key for apps. They need to make sure users can talk to each other in time. Modern apps rely on real-time communication to work well. It's essential, for an user experience. Using modern apps. The old method of checking for updates has its downsides. Users want an experience when using an app. A responsive user interface is crucial when building web stuff. You need to handle notifications to many users quickly and smartly. Real-time communication is key for apps. They need to make sure users can talk to each other in time. Modern apps rely on real-time communication to work well. It's essential, for an user experience.

Related Work

Previous systems used to work like this:

- They had polling-based notification mechanisms
- They had alert systems that did not give you updates right away

Nowadays systems are different. They use:

- WebSockets so you can get information in time
- Event-driven architectures so they can handle a lot of users at the same time

This project is better than the old systems because it combines real-time updates with a frontend interface that works really well. The real-time updates, in this project make it very useful. The project uses real-time updates to make the system more efficient.

Research Methodology

The development process is, like this:Requirement analysis is the step. We do system design using event-driven principles. Then we work on frontend development to make the user interface update dynamically. After that we do integration with the backend services. The development process also includes testing and evaluation of the system. We have to do testing and evaluation to make sure the system works properly.

Phases of Event driven notification broadcasting service Web development

1. Event Identification Phase

The system looks for all the things that can make it send out notifications. These things can be things like getting a message or when someone does something on the site. The system also looks for updates or when something is confirmed. It even looks for reminders that need to be sent out. When we are designing the parts of the site that users see it helps to know what these events are. This way we can make sure the site can show all the kinds of notifications in a good way.

2. Event Detection Phase

The backend of the system is always watching what is happening. Finds out when one of these events happens. The frontend of the system gets ready to get these events through connections that let it talk to the backend all the time. This way the frontend can get the events away.

3. Data Processing Phase

When the system finds out that an event has happened it does a things. It collects the information it needs. It makes the notification that will be sent out. It puts the information into a format that the frontend can understand. The people making the frontend have to make sure it can understand and show this information in a way.

4. Notification Generation Phase

In this part the system makes a message based on what happened. It also decides how important the notification is and what kind of notification it is. The frontend is in charge of making the notification look good. It decides what colors and pictures to use. It also groups the notifications so they are easy to look at.

5. Event Broadcasting Phase

The system sends out the notification to the users away using special technologies. This way all the users get the update at the time without having to reload the page.

6. Real-Time Delivery Phase

The notifications get to the users device away. The frontend has to keep talking to the backend. It has to get the notification information. It has to be able to handle a lot of updates at the time.

7. User Interface Rendering Phase

This is the important part, for the people making the frontend. They have to make the notifications show up on the screen in a way. They can use pop-ups or special panels to show the notifications.

8. User Interaction Phase

The users can do things with the notifications. They can click on them to see information. They can make them go away. They can even go to pages from the notifications. The frontend has to make sure all of this works smoothly. It has to make sure the site looks good on all devices. It has to make sure all the interactive parts work well.

9. Logging and Storage Phase

The system saves all the notifications. This way the users can look back at them. The system can also use them to see how things are going. The frontend might give the users a way to look back at their notifications. It might even let them search for notifications.

10. Performance Monitoring Phase

The system checks how well it is doing. It looks at how fast It looks at how the notifications are being delivered. It looks at how fast the site's They have to make sure the updates do not take long.

11. Feedback and Optimization Phase

The system uses what the users say to make things better. It uses their feedback to make the site look nicer. It uses their feedback to make the notifications better. It uses their feedback to fix any problems that come up.

Project Development

1. Requirement Analysis Phase

We need to figure out what users want when it comes to getting notifications away.For the Project Development we have to identify what kind of events will trigger these notifications like messages or alerts or updates. We also have to think about how the Project Development will look and feel to the user.

2. System Design Phase

Now we design the part of the Project Development that users will see.We want to make sure the Project Development looks good on all devices so we plan the layout.We decide how the different parts of the Project Development will talk to each other using something called WebSockets for the Project Development.

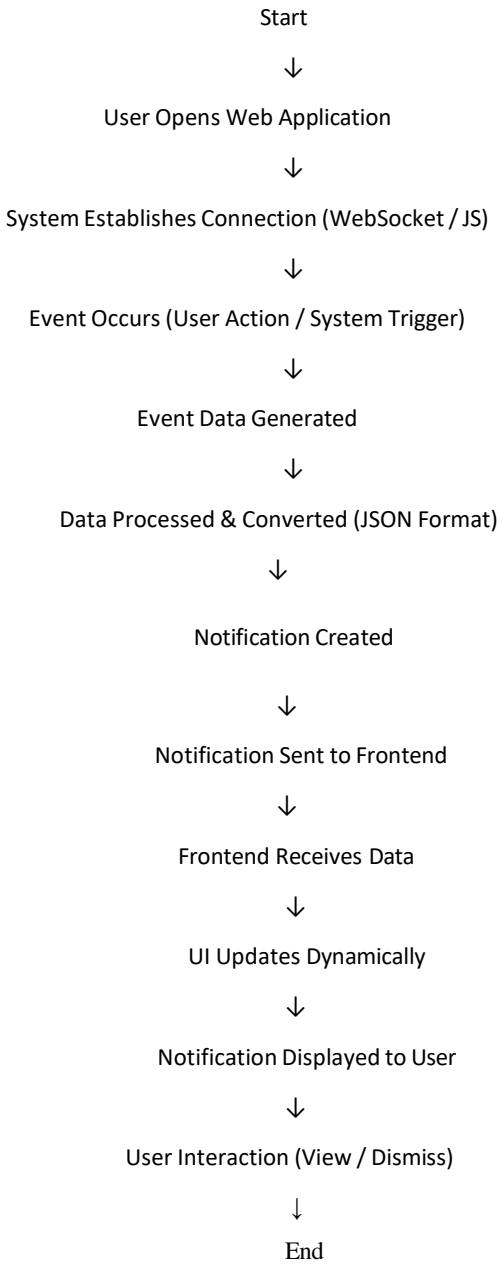
3. Implementation Phase

We start building the user interface for the Project Development using tools like HTML, CSS and JavaScript.Then we set up the WebSocket connection for the Project Development.We make sure the notifications show up correctly and on time for the Project Development.

4. Testing Phase

We check if the user interface for the Project Development works well and looks good.We make sure the updates happen in time, for the Project Development.We test the Project Development on browsers to ensure it works everywhere.

➔ Related flow



Data Preprocessing and Analysis

We do Descriptive Analysis.This is where we analyze how often we send notifications.We also do Correlation Analysis.This means we study how events affect user engagement.Then we do Feature Selection.This is where we pick the things that we want to include in our notifications.Finally we do Visualization. This means we display our data using charts and other things on the user interface dashboards.

1. Collecting Information

We need to gather information, about how users interact with our system. This means we have to identify what events trigger actions.

2. Data Cleaning

We have to remove any invalid data that we do not need. This is important to ensure that our data is consistent.

3. Data Transformation

Text Processing is also important.We have to format notification messages so they make sense to users.Normalization is another step.We have to standardize our data so it looks good on the user interface.

4. Data Analysis

We do Descriptive Analysis.This is where we analyze how often we send notifications.We also do Correlation Analysis.This means we study how events affect user engagement.Then we do Feature Selection.This is where we pick the things that we want to include in our notifications.Finally we do Visualization. This means we display our data using charts and other things on the user interface dashboards.

Outcome

When we look at the outcome of data analysis we get a lot of information about student behavior and course demand and admission trends. This information, about student behavior and course demand and admission trends really helps institutions make their admission strategies better and also improves the enrollment rates of the institutions.

The system gives you updates away with real-time notification delivery.This helps to get user engagement from the people who use it.It also means there is a server load which is really good.The people who use the system have a frontend experience when they are, on the website or app.

Proposed Algorithm

The system is going to use an algorithm that deals with student enquiries and turns them into admissions that are confirmed.

Step 1: When an event happens we need to do a things. First we have to detect that the event occurrence has taken place.

Step 2: Then we generate a notification message for the event occurrence.

Step 3: Next we send the data for the event occurrence via WebSocket.

Step 4: After that the frontend receives the data for the event occurrence. **Step 5:** We then render the notification, for the event occurrence dynamically. **Step 6:** Finally we log the activity of the notification for the event occurrence.

Proposed Solution

The system uses a things to work well.It has event-driven backend logic. When something happens to the system the backend does some things. The system reacts to events. The system talks to the users browser using WebSocket communication. This helps the user. The system uses frontend rendering. This is good for the user. When something changes the system updates the page away. The system feels fast and easy to use. The system gives the user updates without refreshing the page. The event-driven backend logic, WebSocket communication and frontend rendering make this possible. The system uses these to work The event-driven backend logic is really important. It helps the system respond quickly. WebSocket communication helps the system talk to the users browser. That's necessary. Frontend rendering updates the page, for the user. That's good.All these things together make the system really fast. The system works because of event-driven backend logic, WebSocket communication and frontend rendering.

Main Parts of the System

1. Finding the Right Freelancer

The system helps people find a freelancer for their needs. It matches users with services from freelancers. This is really helpful. The system finds a freelancer for you.

2. Suggesting Projects to Freelancers

The system suggests projects to freelancers. These projects are ones they might be interested in. It shows them recommendations. These recommendations are for them. They see projects that freelancers think they can do.

3. Helping with Bids

The system keeps users updated on their bids. It sends them notifications. These notifications are when there are updates on a bid. Users know whats happening with their bids. They get updates, on bids.

4. Stopping Fraud

The system helps keep everyone safe. It watches out for activity. It alerts users to problems. This helps protect both freelancers and users. The system stops fraud. It keeps users safe.

Steps of Data Preprocessing:

Steps of getting the data ready

- First we need to collect the data.
- Then we have to clean the data.
- After that we do the data transformation.
- Next we do the data integration.
- Finally we select the features of the data.

The main steps involved in data preprocessing are described below.

1. Data Collection

We need to gather data from places like what users do and things that happen in the system, logs and databases. For this project data is things like event triggers which're like messages or alerts and information about users that we need to send them notifications. We are looking for Data Collection from sources.

2. Data Cleaning

In this part we get rid of Data that we do not need Data that's the same or Data that is wrong. This helps make notifications better by removing Data that's not good or not complete. We want to make sure our Data is accurate and useful for sending notifications.

3. Data Transformation

We take the Data. Make it into a format that we can use, like JSON. This means we make notification messages look nice make sure values are the same and get the Data ready so it can be used and shown on the website or app. Our goal is to make the Data useful for notifications.

4. Data Integration

We take Data from places like databases, APIs or services. Put it all together in one system. This way we have all the information we need in one place to make sure notifications are accurate. We are talking about Data Integration to make our system better.

5. Feature Selection

We pick the things like what kind of event it's when it happened who the user is and what the message says. This helps us make the system better by using the Data we need to make notifications work. We are selecting features, from the Data to make our notifications more effective. This is part of the Data process.

RESULT EVOLUTION AND ANALYSIS

The **Event driven notification broadcasting service Web development** gives us results that show it really helps with student enquiries and tracking admissions. It also sends out follow-ups on its own. We get reports and dashboards from the **Event driven notification broadcasting service Web development** that help us look at trends in admissions and which coursesre popular. The **Event driven notification broadcasting service Web development** also shows us how many students actually join after applying. This information, from the Student Admission Management System helps institutions make decisions based on what the data says and make the admission process better.

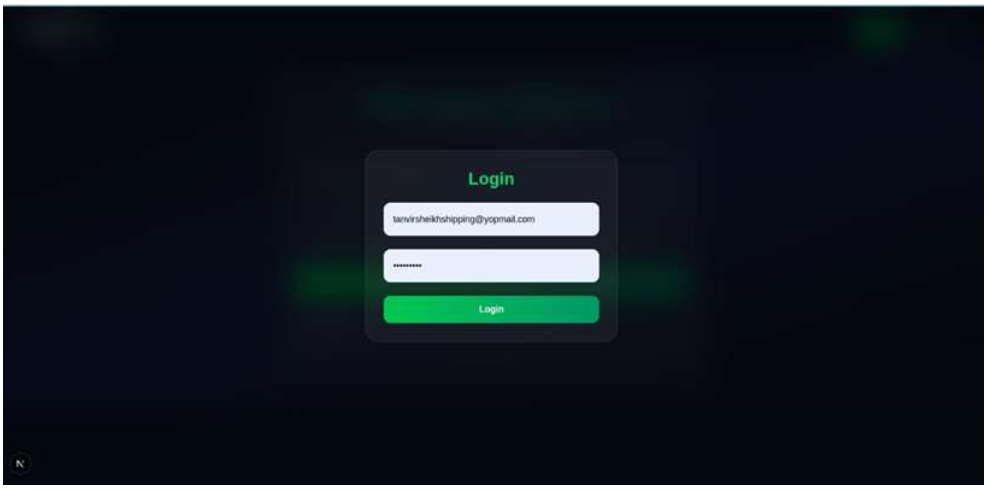


Figure 1: User Login Interface of the System

The image shows the Login Page of the **Event driven notification broadcasting service Web development** . This is where people who are already signed up can put in their information to get into the system. The page has spaces for the users email address and password. After that there is a Login button that checks who the user is and lets them into the system

dashboard.

The way it looks is modern and easy to use. The background is dark. The login part is in the middle, which makes it easier to see and use. This is good because only the right people, like administrators or the people who handle admissionss in the **Event driven notification broadcasting service Web development** . The **Event driven notification broadcasting service Web development** has a system to make sure that only authorized users can get in.

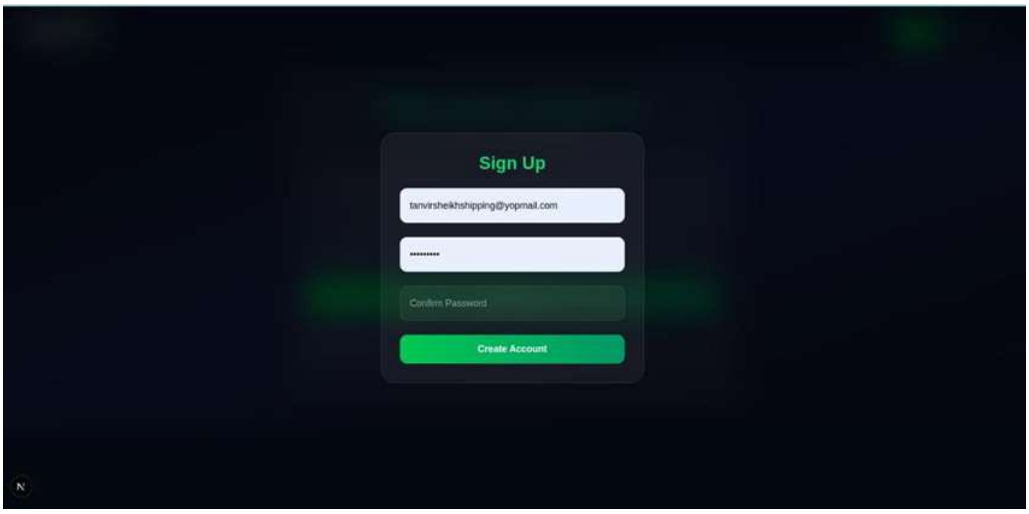


Figure 2: User Registration (Sign Up) Interface

The picture shows the Sign-Up or Registration Page of the **Event driven notification broadcasting service Web development** . This is where new users can make an account so they can use the system. The form has spaces for users to put in their email address, password and the password again to make sure everything is correct.

When users have put in all the information they can click the Create Account button to sign up for the system. The system checks that the password and the password again are the same before it lets users make an account. The **Event driven notification broadcasting service Web development** Sign-Up or Registration Page is really important because it helps new users join the system in a way and take care of things related to admission.

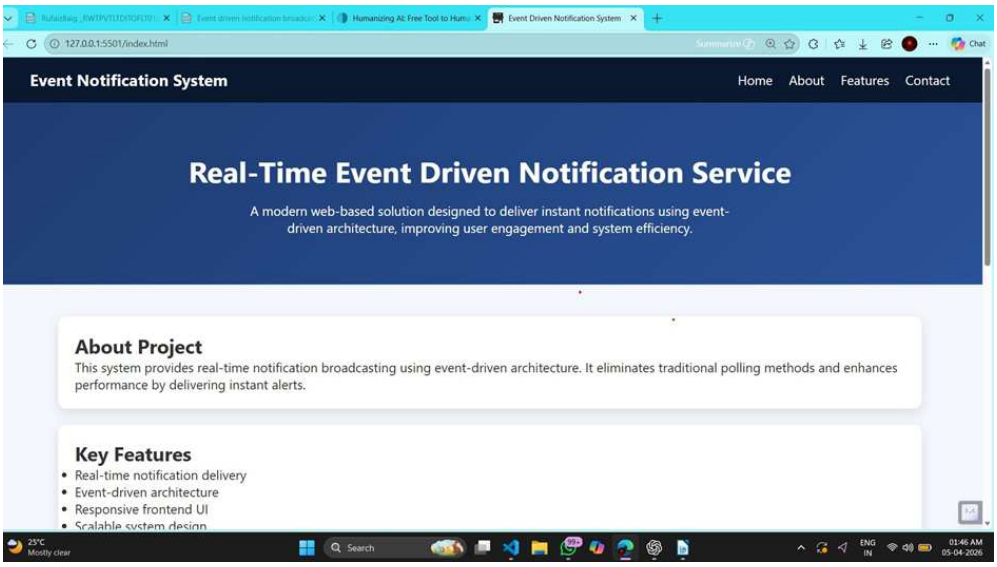


Figure 3: Homepage

The Home Page of the Event-Driven Notification Broadcasting Service is where you begin. This is the place where you find out what the Event-Driven Notification Broadcasting Service is about and what it can do for you. The people who created the Event-Driven Notification Broadcasting Service wanted to make sure it looks good and is easy to use.

At the top of the page there is a bar that helps you get to parts of the Event-Driven Notification Broadcasting Service. You can go to the Home, About, Features and Contact sections of the Event-Driven Notification Broadcasting Service. This makes it easy to get around and use the Event-Driven Notification Broadcasting Service.

Under the bar there is a section that tells you what the Event-Driven Notification Broadcasting Service system is for. It has a title and a short description that explains why getting notifications in time is important for the Event-Driven Notification Broadcasting Service. This part of the page looks nice. Helps you understand what the Event-Driven Notification Broadcasting Service system does.

The main part of the home page is broken up into sections.

- The About Section of the Event-Driven Notification Broadcasting Service talks about what the Event-Driven Notification Broadcasting Service system's how it works. It says that the event-driven architecture makes it possible to get notifications quickly and makes the Event-Driven Notification Broadcasting Service system work better than ways of doing things.
- The Features Section of the Event-Driven Notification Broadcasting Service lists what the Event-Driven Notification Broadcasting Service system can do like give you updates on time work with a lot of users and look good on devices. This helps you see what the Event-Driven Notification Broadcasting Service application can do.
- The Notification Simulation Section of the Event-Driven Notification Broadcasting Service shows you how the Event-Driven Notification Broadcasting Service system works. You can make something happen by clicking a button. It will show you notifications on the screen right away. This is like a test to see how the Event-Driven Notification Broadcasting Service system works in the world.

The home page of the Event-Driven Notification Broadcasting Service also has a section at the bottom that gives you some information about the Event-Driven Notification Broadcasting Service project. This makes the Event-Driven Notification Broadcasting Service application look more professional. The people who made the home page of the Event-Driven Notification Broadcasting Service used HTML, CSS and JavaScript to make it. They wanted to make sure the Event-Driven Notification Broadcasting Service home page

looks good, on screen sizes is easy to use and understand can update the content without reloading the page is nice to interact with

The Home Page of the Event-Driven Notification Broadcasting Service is very important. It helps you understand what the Event-Driven Notification Broadcasting Service system is and how it works. It also makes sure you have an experience when you use the Event-Driven Notification Broadcasting Service application. The Home Page of the Event-Driven Notification Broadcasting Service is where you start. It is what makes your first impression of the Event-Driven Notification Broadcasting Service system.

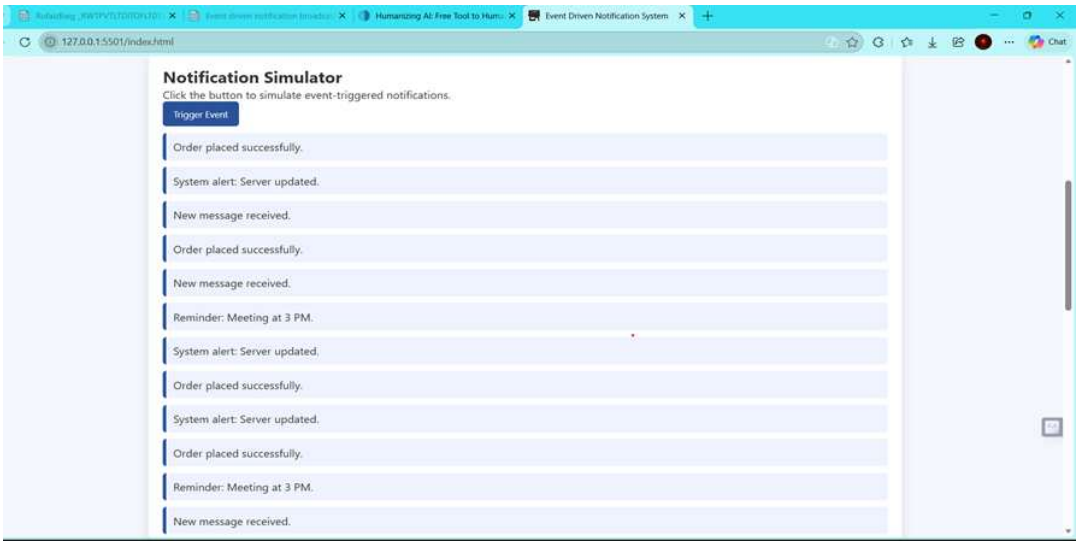


Figure 4: Project Objectives and System Modules Page

The Notification Simulator is an important part of the Event-Driven Notification Broadcasting Service. It helps us see how the system works in time but in a way that we can control. The Notification Simulator shows us how notifications are made and sent to users when certain things happen. This part of the system lets users make things happen by clicking a button. When they do the system makes a notification message away and shows it to the user. The Notification Simulator can also make notifications automatically which is like what happens in life when things are always happening.

The Notification Simulator uses JavaScript to handle events. This is like how some systems can update in time which is a really cool feature.

For people who build the frontend of websites the Notification Simulator is really useful because it shows:

- How to change the webpage without reloading it
- How to show information in real time

The Notification Simulator really helps us understand how the system works, from when something happens to when the user gets a notification. It is like a test version of how we can use this in life with things like WebSockets and server-side event broadcasting. The Notification Simulator is a tool, for learning about the Event-Driven Notification Broadcasting Service and how it can be used in different situations.

Conclusion

The Event-Driven Notification Broadcasting Service shows us how new web technologies can make communication happen in time. When I think about this project as a frontend developer I see how important it is to have an user interface for the website to work well on different devices and for the user to have a smooth experience. The Event-Driven Notification Broadcasting Service really helps by getting rid of waiting times and making people more interested, in the service through notifications that happen away.

References :

1. M. Fowler, *Patterns of Enterprise Application Architecture*. Boston, MA, USA: Addison-Wesley, 2002.
2. G. Hohpe and B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Boston, MA, USA: Addison-Wesley, 2004.
3. L. Richardson and S. Ruby, *RESTful Web Services*. Sebastopol, CA, USA: O'Reilly Media, 2007.
4. M. Kleppmann, *Designing Data-Intensive Applications*. Sebastopol, CA, USA: O'Reilly Media, 2017.
5. D. Chappell, *Enterprise Service Bus*. Sebastopol, CA, USA: O'Reilly Media, 2004.
6. A. Banks and R. Gupta, "MQTT Version 3.1.1," OASIS Standard, Dec. 2015.
7. I. Fette and A. Melnikov, "The WebSocket Protocol," RFC 6455, Internet Engineering Task Force (IETF), Dec. 2011.
8. "Node.js Documentation." [Online]. Available: https://nodejs.org/ . [Accessed: May 13, 2026].
9. "Socket.IO Documentation." [Online]. Available: https://socket.io/docs/ . [Accessed: May 13, 2026].
10. "Firebase Cloud Messaging Documentation." [Online]. Available: https://firebase.google.com/docs/cloud-messaging . [Accessed: May 13, 2026].
11. "Apache Kafka Documentation." [Online]. Available: https://kafka.apache.org/documentation/ . [Accessed: May 13, 2026].
12. "RabbitMQ Documentation." [Online]. Available: https://www.rabbitmq.com/documentation.html . [Accessed: May 13, 2026].
13. R. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Univ. California, Irvine, CA, USA, 2000.
14. M. Richards, *Microservices vs. Service-Oriented Architecture*. Sebastopol, CA, USA: O'Reilly Media, 2015.
15. "MongoDB Documentation." [Online]. Available: https://www.mongodb.com/docs/ . [Accessed: May 13, 2026].
16. "Express.js Documentation." [Online]. Available: https://expressjs.com/ . [Accessed: May 13, 2026].
17. T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.1," RFC 2616, IETF, Jun. 1999.