Web Application using Spring Boot

Project report submitted in partial fulfillment of the requirement
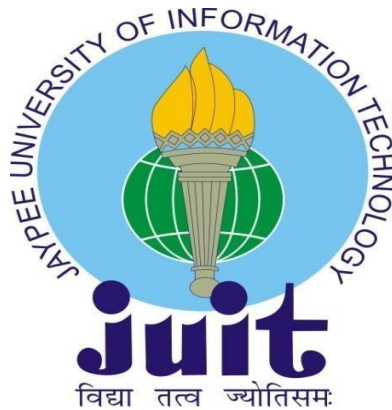for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Ankit Mishra (191395)

Under the supervision of

(Dr. Pardeep Kumar)
(Associate Professor, CSE)

Department of Computer Science & Engineering & Information
Technology
**Jaypee University of Information Technology Waknaghat,
Solan-173234, Himachal Pradesh**

Candidate's Declaration

I hereby declare that the work presented in this report entitled **" Web Application using Spring Boot"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2022 to May 2023 under the supervision of **Dr. Pardeep Kumar** (Associate Professor Computer Scienceand Engineering).
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Ankit Mishra, 191395

This is to certify that the above statement made by the candidate is true to the best ofmy knowledge.

(Supervisor signature)

Pardeep Kumar
Associate Professor
Computer Science and Engineering
Dated:12/05/2023

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT
PLAGIARISM VERIFICATION REPORT

Date: ....

Type of Document (Tick): | PhD Thesis | | M.Tech Dissertation/ Report | | B.Tech Project Report | | Paper |

Name: _____ __Department: _____ Enrolment No _____

Contact No. _____E-mail. _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:
- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)   **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature   **Librarian**

....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

Table of Content

List of Abbreviations

- **MVC -**   **Model-View-Controller**
- **REST-**   **Representational State Transfer**
- **GSP-**   **Groovy Servers Pages**
- **HTTP-**   **Hypertext Transfer Protocol**
- **GET-**   **HTTP Get request methodology**
- **POST-**   **HTTP Post request methodology**
- **GORM-**   **Grails Object relational mapping**
- **HQL-**   **Hibernate Query Language**
- **EDD-**   **Event Driven Development**
- **TDD-**   **Test Driven Development**
- **CRUD-**   **Create Read Update Development**
- **AJAX-**   **Asynchronous JavaScript And XML**
- **XML-**   **eXtensible Markup Language**
- **CDN-**   **Content Delivery Network**
- **HTML-**   **Hypertext Markup Language**
- **CSS-**   **Cascading Style Sheets**
- **JSON-**   **JavaScript Object Notation**
- **API-**   **Application Programming Interface**
- **JWT-**   **JSON web token**
- **SQL -**   **Structured Query Language**
- **URI-**   **Uniform Resource Identifier**
- **URL-**   **Uniform Resource Locator**
- **CLI-**   **Command Line Interface**
- **IDE-**   **Integrated Development Environmnet**
- **UI-**   **User Interface**
- **JS-**   **JavaScript**

- **UDP-**   **User Datagram Protocol**
- **TCP-**   **Transmission Control Protocol**

LIST OF FIGURES

Abstract

On 21 June 1948, For the very first time a piece of program was successfully held in an electronic memory and was executed successfully. After this event, advancement in software development started and multiple software development architectures were introduced for different purposes. And with addition of new programming languages and with multiple iterations of software design architectures, some of the architectures got much more useful than others.

The entire layout and structure of a software system are implied by the term "software architecture." It covers all of the vital components, their interrelationships, and the rules and requirements that direct their behavior. Software development employs a wide variety of architectures, each of which has advantages and disadvantages of its own. In this article, we'll examine some of the most popular architectural styles, as well as their advantages and disadvantages.

Initially, when only only goal of building software was to just do it work then monolith software architectures were used mostly which was mostly messy and was very complex and did not follow any specific way to write code but as coding and programming became more and more popular, more people found new and much better ways to solve the problems. Thus, many new "**Software Development Architectures**" were invented, each with their own merits and demerits.

A software system's software architecture describes the general layout and design of the software system. It includes all necessary elements, their connections to one another, and the specifications on how they should work. Many different architectures are used in software development, each of which has benefits and drawbacks to take into account. In this post, we'll look at some of the most common architectural styles and talk about their benefits and drawbacks.

The framework specifies the system's quality characteristics, such as efficiency, adaptability, dependability, and privacy, and offers rules for making sure that these characteristics are met throughout the development process. This ensures system quality. Reuse is made easier by a designed effectively software architecture, which also speeds up and lowers the expenditures of creating software by allowing for the reuse of parts and modules in various projects. There are multiple steps when we create or decide which architecture to use, we use certain conditions or requirement such as Architectural Synthesis, which is basically first iteration of ideas while creating architecture to solve our problem. After that we have Architectural evaluation where we make further improvements in our architecture. This evaluation can occur at any step of planning or designing the architecture. And to ensure quality of software architecture, this is step is very crucial.

CHAPTER - 1 INTRODUCTION

1.1 Introduction

While building a software in modern times, developers have a choice from a wide array of Languages, Frameworks, Databases as well as different type of architectures needed to develop their software according to the needs.

Some of the most architectures are listed below :

Monolith Architecture :

When creating software using the traditional approach, also referred to as mono-lithic architecture, each component of an application must be integrated and provided as a separate unit. This approach makes it difficult to manage and expand a single, large codebase due to the interdependencies between the storage, user interface, and application coding. Despite its flaws, monolithic architecture has been used effectively in many applications and is still an option in some use situations.

One advantage of monolithic architecture is its simplicity. Since every part of the programme is integrated into a single codebase, it is straightforward to understand the program's architecture and modify the code. Therefore, monolithic design is suitable for small-scale applications or projects with a distinct objective.

Additionally, monolithic architecture can be used to build systems that require an elevated degree of security or durability since each part of the programme can be meticulously reviewed before distribution.

Additionally, monolithic designs have the advantage of functioning well even programmes with very little viewership. Because each programme

element is deployed as a single unit, monolithic design may be less wasteful of resources than alternative approaches. Small-scale applications may profit from this as it will likely lead to quicker response times and cheaper server expenses.

Monolithic architecture faces a few substantial drawbacks despite its benefits. The difficulty in scaling monolithic programming is one of their key drawbacks. Scaling necessitates duplicating the whole stack because all of the application's components are interrelated. Performance problems that are expensive and time-consuming and unavailability during scalability events can result from this.

Monolithic design also has the drawback of being challenging to sustain as an application expands. It could get more challenging to modify the code without adding errors or erasing already-existing functionality as the codebase grows. Because of this, adding new features or fixing bugs may be difficult, especially if the application has been created by a team.

Despite these shortcomings, monolithic design has been effectively used in many applications. One example is the free content management system WordPress. The PHP-based WordPress program's monolithic architecture offers a reliable and flexible platform for content management. WordPress is a well-liked alternative among businesses of all sizes, and millions of websites have been created using it.

And finally, a frequent technique for developing programming that excels at specific activities or applications is monolithic architecture. It is simple to understand and appropriate for applications that require an exceptionally high degree of security or dependability. However, as the software grows, expanding and regulating it can be difficult, making it less suitable for large-scale applications.

Though monolith architectural has many limitations, it is widely used in many applications even in these modern times. One of the biggest biggest application that uses it successfully till date is WordPress.

Client-Server Architecture :

The client, which makes requests from the server, plus the server, that responds to those requests, make up the system. Client-server architecture is a typical design for developing software. The client device is in control of the customer's interface in this design, which is founded on the concept of assigning responsibility, while the computing system is in responsibility of the company's logic as well as information storage.

In a client-server architecture, the client sends an inquiry to the underlying server, while the server processes and answers. A client can be a desktop machine, computer, smartphone, or tablet as long because it has connectivity with the server. Any gadget which is capable of sending and receiving requests, such as a server for the internet, databases server, or application server, can function as the server.

Scaling is easy with client-server architecture, which is one of its benefits. According to the volume of traffic it receives, the server can simply scale up or down because it manages requests and stores data.

This implies that the server may be scaled up to handle the increased load as the system's user base expands. Similar to this, if fewer users are logging on, the server's capacity may be lowered down to save resources.

Another advantage of client-server architecture is enhanced security. The server may contain security features like access control, encryption, and authentication to prevent unwanted access to private information because it manages the two components of the organization's logic and

the information storage.Web applications frequently use a client-server design, which means the user's device is a browser for the internet and the server that hosts the application is a web server. In this scenario, the client sends HTTP requests to the server, which returns HTML, CSS, and JavaScript replies that the browser can utilise to render as user interfaces. Since it makes it simple to install online programmes and enables customers to connect to the software from every gadget with a web browser, this design is very well-liked.

Online gaming is another instance of client-server architecture where the server controls the playing scene and player interactions while the client is a computer or gaming device.

With this situation, the client asks the server for permission to alter the environment of the game in order that it may move the player, engage in player interaction, and perform other tasks via a user interface that is graphical.

Client-server architecture, in general, offers a strong and adaptable foundation for developing software that is simple to scale, has improved security, and efficiently communicates among clients and servers. This concept is perfect for usage in online games, web applications, and other systems that call for regular interaction across the client and the server.

Test-driven Architecture :

Test-driven development (TDD), a methodology for writing software, places a high emphasis on the implementation of automated tests before writing any code. By initially creating a test case, programmers use this method to create a particular piece of functionality. The developer creates the smallest amount of code required to pass the test once it has been created, then changes it if needed to enhance its design.

Although TDD can be used independently, it is often implemented in

tandem with other development approaches like agile or extreme programming. TDD's primary tenet is that it aids programmers in creating more modular, comprehensible code that is simpler to preserve over time. Here, tests of the project are written even before the code of the application is written, and after the tests are written, we write application code keeping those tests in mind, and we keep using those tests in between development stages to check if our development is on correct path or not.

One of the key benefits of TDD is the ability to identify errors prior to the development process. Developers can find bugs and faults early on by building tests before writing code, lest they become more complex and time-consuming to resolve. TDD also motivates programmers to create more flexible and code that can be reused, which may eventually conserve time and effort.

The communication among developers and other interested parties may be facilitated via TDD. Developers can make sure that everyone understands what the system should do by writing tests that specifically describe the intended behaviour of the system. This will make it easy to prevent misunderstandings and guarantee that the system satisfies the needs of all stakeholders.

TDD can be helpful for creating an online application that must be both extremely responsive and scalable. TDD provides developers with the reassurance that the software runs smoothly and efficiently. TDD can be used to find bugs and mistakes that could lead to application crashes or other failures while an application is under load.

TDD is a powerful methodology that can improve computer system efficiency and maintainability on a broad scale. TDD can assist developers in creating better software more rapidly and with fewer errors and defects by putting an emphasis on software testing

automation and modular architecture.

Even though this architecture has its own shortcomings, but this in some niche use cases, TDD approach shines brighter than the other type of modular architectures.

Event-driven Architecture :

The event-driven architecture (EDA) model of software development significantly emphasises the use of events as the major conduit for data transfer and interaction between software components. When an event is generated by a third party, such as input from users, detectors, or other software systems, the EDA system reacts to it. When these things happen, the system can respond quickly to input from the user and change its course to start performing specific tasks or functions.

The capacity of the event-driven architecture to easily handle complex, dynamic systems is one of its main advantages. To create systems that react rapidly and successfully to changing conditions and requirements, events might be employed as the main route of communication.

When designing distributed systems and microservices, EDA is frequently utilised since these systems need to be able to adapt to alteration in demand, network conditions, and other variables.

The application of event-driven messaging systems in financial trading platforms is one illustration of EDA in action. These systems need to be capable of handling millions of transactions per second while also being adaptable to changes in the market and other outside variables. The platform can process incoming data quickly and effectively, send out the required responses, and keep up with the hectic world of financial trading thanks to its event-driven architecture.

The fact that EDA is horizontally scalable makes it excellent for cloud-based applications. Systems can be built to rapidly grow up or down in

response to variations in demand by using event-driven communications. It renders it feasible for drivers to manage unexpected problems or abrupt surges in traffic without the assistance of an operator.

However, there are several significant disadvantages to EDA. Designing and administering event-driven systems is one of the major challenges. The event-driven architecture cannot handle events accurately and effectively without careful design and preparatory work. Actions can be initiated from a variety of inputs and can communicate intimately with other system components, making event-driven systems difficult to debug and monitor.

In general, event-based architecture is a useful method for building responsive, dynamic software systems. It works especially well with cloud-based apps, microservices architectures, and distributed systems. The system must be carefully designed and planned in order to operate appropriately and successfully.

REST Architecture :

REST (Representational State Transfer), a software architecture paradigm, is used to create web-based applications that communicate via HTTP/HTTPS. It offers a clear procedure for creating APIs for the internet that may be used by a variety of clients, such as smartphones and web browsers.

Clients can communicate with assets in a RESTful framework by using GET, POST, PUT, and DELETE standard HTTP methods. Unique URIs are used to identify resources in RESTful architectures. The architecture places a lot of focus on statelessness, which means each request comes with all the information the server needs to process it. RESTful services are therefore reasonably scalable and simple to cache.

The simplicity and utility of REST are two of its main advantages. It does not require any particular libraries or frameworks and can be constructed with any programming language or environment that supports HTTP. REST is also quite flexible and may be used to build a wide range of applications, from simple CRUD operations to complex relationships and processes.

However, there are certain limitations to REST. It can be difficult to decide on sophisticated data models and to manage transactions involving various resources. Because REST is based on HTTP, it may also be less efficient than alternative protocols for specific types of interactions, such real-time messaging.

REST is a popular and incredibly flexible architecture for creating web APIs and is a solid option for a great deal of software programmes that must deliver information or amenities over the internet.

MVC Architecture :

The well-known MVC software development paradigm divides an application into three interdependent parts: Model, View, and Controller. By encouraging the separation of concerns, this design is used to enhance the excellence, reliability, and reusing of code. The Controller operates as the interface among the Model and View to handle user input and update the Model, whereas the Model contains data and the company's logic and the View provides the presentation layer.

Example:

Web application development is one field whereby MVC architecture is being successfully applied. In this case, the Model stands for the application's data, the View for the way it looks, and the Controller is responsible for its user input. As an example, think of an online store.

The Controller would deal with the user's inquiries for queries and the procedure for adding items to the shopping cart. While the Model keeps every detail about those goods and their data, the View shows the user each product and their details.

MVC architecture is sometimes preferred over alternative designs such as a monolithic structure or microservices architecture. MVC architecture offers greater flexible and adaptable than monolithic layout, yet it is too complex to deploy for small programmes with limited scalability. MVC design, on the other hand, is flexible, scalable, and promotes code reuse, rendering it suitable for an extensive variety of applications. Furthermore, the MVC architecture becomes simpler to maintain and test than other solutions.

Microservices Architecture :

Microservices architecture was an application development framework that uses small, self-contained services to generate huge systems. Each service operates as a separate process and communicates with its peers via lightweight protocols, the vast majority of that being HTTP resource APIs. This design's key advantage is that it provides scaling, flexibility, and tolerance for failures in massive amounts application distribution.

Advantages of Microservices architecture:

Scalability: Changes to one service have no effect on the others because each microservice is built separately and has its own API. This enables modifications and upgrades to a particular service to be made without affecting the entire system.

Flexibility: Since each microservice is developed independently and has its own API, changes to one service do not affect the others. This makes it easy to make changes and updates to a single service without impacting the entire system.

Fault tolerance: If one of the services fails in a microservices design, other parts of the network will continue to operate. This is due to the fact that each service is separated and may be restarted or replaced independently.

Faster development and deployment: Because every microservice can be created and delivered individually, new features and upgrades may be delivered more quickly. This facilitates responding to shifts in consumer demands or market conditions easy.

Better scalability of teams: Teams of fewer people can be in charge of particular services with a microservices design, allowing them to function better efficiently and autonomously. This enables more interaction, quicker decision-making, and faster development.

Microservices design is superior to alternative architectures in the following cases:

Netflix is a well-known example of microservices architecture. Spotify has more than 200 million customers in over 190 countries and provides more than one billion minutes of streaming content each week. With such a massive user base and complicated business requirements, Netflix required a scalable infrastructure that could react to changing market conditions.

Netflix continues to be enabled to continually develop and enhance its services since deploying microservices architecture in 2009. When Netflix opted to reach global markets, it had to cope with a wide range of spoken languages, currency pairs, and payment systems. Netflix was able to swiftly develop and launch additional services to suit these requirements due to its microservices methodology.

For fault tolerance, Netflix also employs a microservices model. If a certain service fails, the rest of the system can continue to function,

letting users continue viewing films and TV episodes without interruption.

Microservices architecture has become a more and more common strategy for large-scale, complicated applications due to its versatility, scalability, and fault tolerance. Individual independence in service creation and deployment facilitates speedier development and innovation. Fault acceptance and scalability are improved by using lightweight communication strategies and service isolation. Netflix is a great example of a company that has effectively adopted microservices design, allow it to react to shifting market conditions and evolve swiftly.

1.2 Problem Statement

The problem statement is to design an application to clearly show the advantages of Microservices Architecture.

Requirements of Web app :

We need to develop a link sharing application which could be used for sharing usefullinks/documents amongst a group of subscribers. Users can create new topics or subscribe to existing topics. A topic can be either private or public. A public topic is visible/open for subscription to every user. A private topic can be subscribed only through an invitation sent by an existing subscriber. The application should provide a solution to following stories:

1. A user should be able to login.

2. A new user should be able to register. An active valid user should be able to login with correct credentials.

3. A user should be able to reset his/her password by clicking on the forgot password link.

4. User can create a new topic and he will be automatically subscribed to it with seriousness(SERIOUS, CASUAL, VERY_SERIOUS) of Very Serious. The topic can be private or public. Name of the topic should be unique per user.

5. User can subscribe to an existing public topic.

6. User can specify his/her seriousness to a particular topic.

7. Subscribed users can send invites for a public or private topic.

8. The user should be able to browse all the public topics.

9. The user should be able to add a resource to a subscribed topic.

10. The user cannot be deleted.

11. The user should be able to mark a resource as read/unread.

12. Only a Creator of a resource or admin can delete a resource.

13. Only a Creator of a topic or admin can delete a topic. Its resources should also be deleted irrespective of the ownership status or resources.

14. User can rate a resource. Most subscribed topic will be a trending topic.

1.3 Objectives

The objective of the project titled Microservices Architecture with Spring Boot is to design and develop a microservices-based architecture for a software application using Spring Boot framework. The project aims to demonstrate the benefits of microservices architecture over traditional monolithic architecture by providing a modular, scalable and

resilient system that can handle a large amount of traffic and data.

The project will focus on building a web application that provides various functionalities to the users. The application will consist of multiple microservices, each responsible for a specific functionality. The microservices will communicate with each other through well-defined APIs, and each microservice will be independently deployable and scalable.

The key objectives of the project are:

1. Design an Application using Microservices Architecture: The first objective of the project is to design a microservices-based architecture that is modular, scalable, and resilient. The architecture will be designed to support the separation of concerns, allowing each microservice to be independently developed, tested, and deployed. The design will take into consideration the key principles of microservices architecture, such as loose coupling, high cohesion, and autonomy.

2. Develop Microservices: The second objective of the project is to develop microservices using the Spring Boot framework. Spring Boot provides an easy-to-use, lightweight, and opinionated approach to building microservices. The project will leverage the features of Spring Boot, such as auto-configuration, embedded servers, and dependency injection, to build efficient and scalable microservices.

3. Implement API Gateway for necessary routes: The third objective of the project is to implement an API Gateway that will act as a single entry point for all the microservices. The API Gateway will provide various functionalities such as authentication, routing, and load balancing. The project will leverage Spring Cloud Gateway, a lightweight API Gateway built on top of Spring Boot, to implement the API Gateway.

4. Implement Service Discovery: The fourth objective of the project is to implement service discovery using Spring Cloud Netflix Eureka. Service discovery is a critical component of a microservices-based architecture as it allows the microservices to discover and communicate with each other dynamically. Eureka provides a simple and efficient way to implement service discovery.

In conclusion, the objective of the project titled "Microservices Architecture with Spring Boot" is to design and develop a microservices-based architecture for a web application using Spring Boot based framework. The project aims to demonstrate the benefits of microservices architecture over traditional monolithic architecture.

1.4 Methodology

To make the required Map, we need to create following tables in our domain classes so that we
can make our web application according to given requirements. We need to map our tables carefully to make sure that when one record in a table is deleted then we also all the records on other tables that are dependent on deleted record.
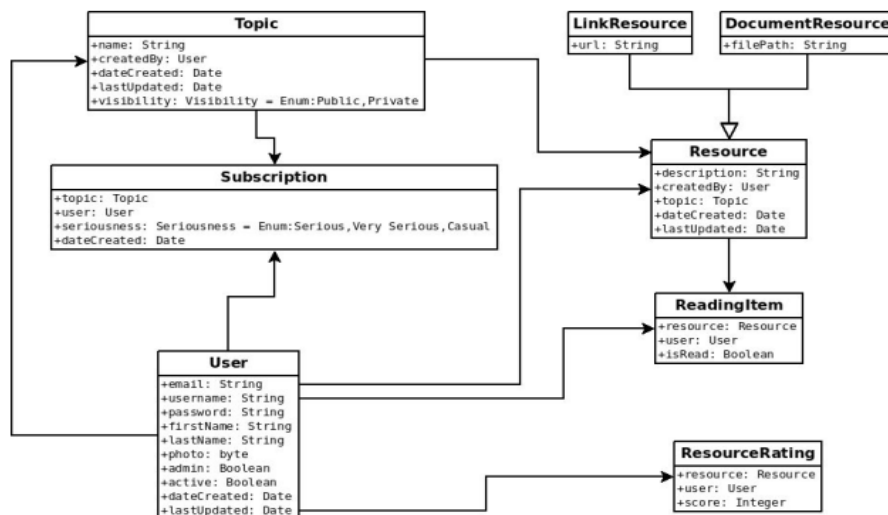

Fig 1.1 : Domain Classes Structure

14

Hence, As shown in above table, mapping of our tables will be as shown :

User Table → has many [Topics, Resources, ReadingItems, ResourceRatings, Subscriptions]

Topics → has many [Resources, Subscription]

Subscription → belongs To [User, Topics]

Resource → has many [Reading Item]

Resource → belongs To [Topic]

Reading Item → belongs To [Resource]

Resource Rating → belongs To [User]
Link Resource → extends Resource

Document Resource → extends Resource

To create a domain class inside our application :
    → To go terminal
    → Write : grails create-domain-class user
    → Above command will create a user class in domains directory in our application,
    → Hibernate will create a mapped table in database, that matches user domain class.

Fig 1.2: User domain

Fig 1.3: Topic domain

16

Fig 1.4:Subscription domain

Fig 1.5:Resource domain

Fig 1.6:ResourceRating domain

Fig 1.7:ReadingItem domain

Fig 1.8:DocumentResource domain

Fig 1.9:LinkResource domain

After that, the URLs are routed using a number of controllers, and inside of every controller, we are given the opportunity to build an array of methods (also referred to as actions), which can be then used to carry out a variety of tasks. After building a controller, you must offer services to various controls that comply with the requirements.

The service in Grails is an object that divides programming functionality and gives other sections of the app's code a method to access it. To provide corporate logic and integrate with various software components such controllers and data access objects.

1.5 Organisation

Five chapters make up the substance of this project report. Several research articles that are related to this project activity are included in the literature review, which is detailed in chapter 2 after this introduction chapter. The Second chapter provides an overview of the various works on similar domains that was carried out previously by various authors. The third chapter gives an overview of the proposed model about how the web application architecture is specifically designed.

CHAPTER - 2

LITERATURE SURVEY

2.1 Introduction

Subscription-Based Web Application

Subscription-based web apps have grown in appeal in the past decade because they provide developers with a long-term business strategy while giving significant value to clients. Customers may subscribe to the app's contents or services, and the creators may charge an annual subscription fee for ongoing access. Netflix serves as a renowned web programme that operates on a subscription basis. Netflix is an audio streaming provider that charges clients a monthly subscription fee to view its huge library of films and TV series. In contrast, Spotify is a streaming music service that asks for an annual fee to having access to its extensive song library. These apps have grown in favour since they make it easy and affordable for clients to access information.

Subscription-based internet applications have increased in popularity over the last decade because they give programmers with a strategy for the future while providing significant advantages to clients. Clients may subscribe to the application's contents or services, and the producers may charge a yearly cost for continuous access. Netflix is a well-known web programme that runs on an ongoing subscription basis. Netflix is a multimedia streaming service that charges customers a fee per month to access its massive repertoire of films and TV shows. In contrast, Spotify serves as a music streaming service that charges an annual fee for access to its huge song catalogue. These apps have gained in popularity because they make it simple and economical for clients to obtain information.

Users will be able to subscribe to topics of interest in the proposed system and will be notified when new documents on those topics are uploaded. The concept of topic subscription is well-known on social networking sites such as Facebook, Twitter, and LinkedIn. Users can subscribe to or follow themes in order to receive notifications when new content is published to specific topics. This programme keeps users informed of the most recent trends and news in their area.

The possibility for users to add resources to the topics to which they have subscribed is another feature of the proposed system. This function is comparable to content sharing on social media platforms like Reddit and Pinterest.

Users can share links and documents linked to themes with other users who have subscribed to similar topics. This encourages users to collaborate by sharing information and resources.

The suggested system also includes an administrator user with the ability to delete subjects, resources, and people. This skill is required to ensure the quality and integrity of the platform's content. The admin user has the ability to delete any unnecessary or unsuitable content or individuals, ensuring that the platform remains a trustworthy source of information for its subscribers.

CHAPTER - 3

SYSTEM DESIGN AND DEVELOPMENT

3.1 Introduction

Software architecture for our Microservices based web application is User Login, Topic Creation, Topic Subscription, and Link Resource Sharing and Document Link Resource and some more features.

Users have a place to exchange information and find resources related to various topics thanks to the proposed programme. People can register on the website and contribute their own subjects, to which other users can subscribe. After enrolling, users can add webpages and papers from universities as resources to the themes they've selected to follow. Trending topics and subscriptions will also be shown on the user dashboard, and an admin user will have the opportunity to delete resources, persons, or topics. Users can also add, update, and remove their own content and subjects. Every time a new asset is added to a subscribed subject, a dashboard inbox is also updated.

User Management:

The initial step in system design is the development of a user management system. This will involve processing of passwords, user authoriisation, and login. Users should be able to create new accounts and log in to the platform with those credentials. Passwords should be securely stored using a hashing method. It should be possible to reset forgotten passwords using the system's email confirmation.

Topic Creation:

A user ought to be able to add a new subject after logging in. Every topic must have a name, a description, and tags. By browsing at

prevalent subjects on the overview page, users can find topics that interest them.

Resource Sharing:

Resources must be able to be added by users to the topics to which they have subscribed. For instance, resources could include links to other websites or documents that have been contributed to the site. Users must have the option to edit or remove materials that they have submitted.

Dashboard:

The user dashboard must display the user's subscribed topics, trending topics, and reminders. Alerts must be updated each time an additional resource becomes available to a subject which is currently subscribed to.

Subscription List:

The option for members to subscribe to articles made by other users of the app forms one of the main features of the application. Users can utilise this feature to stay current on resources and information relevant to their interests. A summary of how the list of subscribers feature functions is provided below:

1. Subscribing to a Topic: Users have the option to subscribe to topics they are interested in. On the subject matter page, select the "Subscribe" button to accomplish this.

2. Managing Subscriptions: From their dashboard, users may control their subscriptions. They can quickly unsubscribe from any topic if they continue to want to receive updates by looking at this list of every subject they have subscribed to.

3. Hot subjects: On the dashboard, the application will also show a list of the most popular topics at the moment. The number of subscribers indicates that people are currently interested in these topics.

4. Inbox: Users who have subscribed to a topic will receive a

notification in their inbox whenever a new resource is submitted to that topic. Users will constantly be updated on the latest information and resources relevant to their interests thanks to this.

Implementation Details:

The application will need to keep track of which users have subscriptions to which subjects in order to execute the subscription list feature. A "Subscription" domain class that can be created in Grails will include the following fields to accomplish this:

- user: a reference to a subscribed user - subject: a reference to a topic that is subscribed to Unique subscription identification (id)

A new Subscription object will be created and saved to the database each time a user subscribes to a topic. Similar to this, the associated Subscription object will be removed when a user unsubscribes from a subject.

The application must count the number of subscriptions for each topic and sort them in descending order in order to provide a list of the most popular topics on the dashboard.

This can be done using a SQL query, such as:

```
SELECT topic_id, COUNT(*) AS subs_count
FROM subscription
GROUP BY topic_id
ORDER BY subs_count DESC
```

When server-side polling and AJAX are used together, the application may provide inbox alerts. The server will notify all users who have subscribed to it whenever a new resource is published to that subject. Following that, the new message in the inbox will be shown using client-side JavaScript code.

The app's subscription list function, which enables users to stay up to date on news and information relevant to their interests, is one of its key

features. The application is able to offer a seamless and simple user experience by integrating this feature and making use of Grails and other web technologies.

Trending Topic List :

The subjects that are currently sparking the most user discussion are known as trending topics. A list of subjects can be generated based on the amount of subscribers, views, and user communications by researching user behaviour and using this capability.

For Trending topic list implemention, We will be following logic :

1. Data gathering: We need to know the amount of subscriptions, views, and user interactions for each topic. You can accomplish this by having the application log user behaviour.

2. Data analysis: After acquiring information on user behaviour, we can review it to determine the subjects that are the most popular. Several data analysis techniques, such as clustering and classification, can be used to identify the most popular topics.

3. Making a list of trending topics: Data analysis can be used to generate a list of hot topics. The list may be filtered by subscribers or views and frequently updated to ensure that the most well-liked topics reflect the most current user activity.

The trending topic list is a dynamic feature that changes over time in reaction to user behaviour, so it's vital to keep this in mind. The app will be able to give users an overview of the topics being addressed at the time and allow them to keep up with the most recent lectures by integrating this capability.

Visitors can browse the list of popular themes to discover new topics and resources. Users might discover new resources and gain new knowledge. This may lead to more user interaction and a vibrant user community surrounding the programme.

26

To make the list of popular subjects more helpful, we might also add further information, such the number of resources available for each issue and the number of ongoing debates. People might thus find it simpler to locate and communicate with the communities that are concentrated on the issues that matter to them the most.

In general, including the trending topic list in the application can be beneficial because it enables users to find new resources and topics while giving them a rapid overview of the most popular topics being discussed at the moment.

3.2 GRAILS :

Grails is a flexibility based platform. A variety of applications, including those that leverage microservices, can be created using the Spring Boot Grails framework.

Applications are built using a microservices architecture, which is made up of numerous tiny, autonomous services, each of which is in charge of a specific functionality. Because of correctly specified APIs and protocols, these services can interact with one another and work together to deliver a holistic application experience.

Grails is an appropriate choice for developing microservices because of the tools and capabilities it offers. It allows, for instance, the usage of a number of microservices frameworks that offer tools for creating, deploying, and maintaining microservices-based applications, such as Spring Cloud and Netflix OSS.

Additionally, Grails facilitates the development of small, autonomous microservices that are simple to grow and deploy. Numerous programming languages can be used to create these microservices, and they can communicate with one another via message queues or REST APIs.

The following extra Grails functionalities are essential for developing

microservices:

1. Communication with other systems and services that is quick and easy The modules and plugins offered by Grails allow developers to easily link their microservices to other applications and systems. This provides support for REST, SOAP, and JMS in addition to compatibility for other well-known APIs and protocols.

2. Grails is the ideal technology for building microservices because of its speed and light weight. Its memory footprint is also little. Cloud systems' rapid startup times and relatively minimal memory footprint make scaling up and down straightforward.

3. Support for containerization: Grails offers support for the use of containerization tools like Docker and Kubernetes, which make it easy to package and deploy microservices in environments that use containers.

4. Effective control and direction For managing and keeping track of microservices, Grails provides a number of capabilities, including support for distributed tracing, logging, metrics collection, and visualisation.

Overall, Grails provides a solid and flexible platform for developing microservices-based systems. Thanks to its support for microservices frameworks, lightweight architecture, and simple connection with other systems and services, it is a fantastic choice for developing scalable and dependable microservices-based applications.

Online application development has been sped up and made easier with the help of the Grails web application framework. Groovy, a programming language, serves as its foundation. Since its first release in 2006, it has been maintained by The Grails Association.

The framework's creator, Graeme Rocher, aimed to provide a novel approach to creating web applications by combining the ease of Ruby on Rails with the flexibility and power of the Java Virtual Machine (JVM).

He started working on Grails in 2005, and the initial version was released in 2006.

By using Grails, web application developers may rapidly and simply design apps. Startups and small to medium-sized businesses might tremendously benefit from it because they require initiatives that can be developed and implemented fast.

The following are just a few of the many benefits that make Grails a popular option for web development.

- Convention over configuration: By providing default settings and conventions that developers may utilise without having to manually configure everything, Grails adheres to the "convention over configuration" paradigm.

- Easy integration: Grails makes it straightforward to mix a range of technologies, including databases, templating engines, and security frameworks.

- Rapid development: Grails' many built-in features and plugins allow programmers to quickly design web applications.

- Familiar syntax: Developers are familiar with Groovy, a Java-like programming language that is utilised by Grails.

Only a handful of the many advantages of Grails, which make it a popular choice for web development, are briefly discussed in this article.

- Convention over configuration: Grails adheres to the "convention over configuration" paradigm by offering default options and conventions that developers may use without having to manually configure everything.

- Effective fusion: The Grails framework streamlines the fusing of several technologies, including as databases, templating engines, and security frameworks.

– Quick development Programmers may quickly construct web apps

with Grails' many built-in capabilities and plugins.

- Familiar syntax: Grails uses Groovy, a Java-like programming language.

- Efficient fusion: The Grails framework makes it simple to meld several technologies, including as databases, templating engines, and security frameworks. - a thriving developer community that nurtures, encourages, and develops framework extensions. This suggests that developers may design their apps using the approaches and tools they are already accustomed to. There is a big and supportive developer community for Grails. Developers will find it easier to learn how to utilise the framework and apply it, which will maintain it up to date.

Grails is a dependable and adaptable web application framework with a number of advantages over rival frameworks. It is an enticing option for developers who want to make web applications quickly and easily because of its convention over configuration approach, simplicity of integration with other technologies, and promise of rapid development. It is a great option for enterprises that need to manage a lot of traffic and data because of its performance and scalability. The Grails framework is expected to become more well-known in the upcoming years due to its robust development community and expanding selection of plugins and utilities.

GRAILS GORM :

Grails GORM (Grails Object-Relational Mapping), the foundational component of the Grails web application framework, is a dependable and approachable database access and persistence strategy. This essay will outline the background, salient features, and benefits of GORM.
GORM was initially developed by Graeme Rocher, who also developed

the Grails framework. The primary objective of GORM was to provide a simple and user-friendly means of accessing and persisting data to a relational database without the need for laborious SQL queries or JDBC programming. GORM relies on top of Hibernate, a popular Java ORM framework, to provide a straightforward, high-level API that enables developers to work with relational databases in a more natural way.

Main Features of GORM:

1. Domain-Driven Design (DDD): GORM follows the Domain-Driven Design (DDD) methodology, emphasising the need of building software based on in-depth business domain knowledge. Developers can use GORM to generate domain objects that represent the entities of the business domain and automatically map to database tables.

2. Object-Relational Mapping: The GORM framework includes a powerful ORM layer that immediately transforms domain objects into database tables and vice versa. Developers don't need to write any SQL queries or JDBC code to communicate with the database. Instead, they can use GORM's user-friendly API to save, edit, and remove objects from the database.

3. Querying: Using the powerful GORM Query Language (GQL), programmers
 can construct complex queries on the database. The sophisticated andversatile
 query operations made possible by GQL include filtering, sorting, and aggregation.

4. GORM provides a useful technique for using dynamic finders to query the database. Developers can build searches based on the name of the invoked method by using dynamic finders. This method eliminates the need to manually build complex SQL queries.

5. Transactions: Because GORM supports transactions, programmers can aggregate numerous database operations into a single transaction. By ensuring that all actions are either committed or rolled back

concurrently, this helps to ensure database consistency.

Benefits of GORM:

1. Database-driven applications may be created quickly and effectively by developers thanks to GORM's user-friendly API and powerful ORM features. By removing the requirement for writing intricate SQL queries or JDBC code, GORM speeds up development and increases productivity.

2. Versatility: GORM is quite versatile and works with a variety of databases, including MySQL, PostgreSQL, Oracle, and SQL Server. MongoDB and Redis are supported NoSQL databases, while GORM is a flexible option for creating contemporary online software.

3. Maintainability: Because GORM uses a domain-driven design approach, changing database schemas over time is simple. The domain classes, which are simple to read and edit, define the schema. The automated structure creation and update features of the GORM make managing database schema very simple.

4. Performance: Thanks to its clever caching techniques, effective querying, and enhanced SQL generation, GORM offers great performance. The application's overall efficiency can be increased by drastically reducing the amount of database accesses thanks to the GORM caching features.

5. Integration with Grails: GORM is a crucial component of the web application framework Grails, which provides a lot of capabilities and tools for creating modern web applications. It is simple and quick to construct full-stack web-based applications using GORM and Grails together.

To sum up, Grails The robust and user-friendly GORM framework for database access and persistence provides a variety of capabilities and benefits for creating modern web applications. The GORM's domain-driven methodology, strong ORM capabilities, and flexible query language.

3.3 AJAX

The web development method known as AJAX allows webpages to be modified asynchronously without necessitating a full page refresh. It is also known as asynchronous JavaScript and XML at times. XML, HTML, JavaScript, CSS, and other web technologies are a handful that have been combined. Around the turn of the century, AJAX became well-known for its ability to improve user experience by making websites more engaging and responsive.

AJAX was required because of the limitations of traditional web development techniques, which required a complete page refresh for each client interaction with a website. As a result, web user interaction and intuitiveness reduced while page load times increased.

For instance, a user would need to submit an application and wait for the server to respond if they wanted to see if a username was easily accessible on a registration page. The user experience would be time-consuming and challenging as a result.

With AJAX, which permits asynchronous connections to the server that update certain sections of the web page without adding requiring a full page reload, this issue is resolved. Because adjustments to the page may be made in real time without experiencing any discernible lag, the result is a more flexible and imaginative user experience.

One of AJAX's main benefits is the improvement of web application performance. By delivering asynchronous requests to the server, AJAX allows programmers to get data and modify websites without the need for a full page refresh. The outcome is less time for loading and an enhanced user experience.

Another benefit is the ability to create interactive and dynamic web apps using AJAX. Web designers can construct dynamic, real-time websites with AJAX that don't require refreshing. This enables the creation of web applications that function more like native ones, with fluid

transitions and immediate feedback.

In modern applications, many authentication system's uses AJAX, to tell users if the unique username they want to take for themselves in available or not, In this particular feature, whenever user stops typing in input text field, An AJAX method, sends a request in Database to check whether given database is available or not. And if its not available, we show a warning message to user saying, Please change your username.

AJAX offers several useful application possibilities in addition to its technological benefits. For instance, e-commerce systems usually offer up-to-date details on product availability and pricing. Users of social networking applications can view real-time account modifications and conversation thanks to its utilisation.

Real-time collaboration is essential for many web-based productivity tools, such as Google Docs, which frequently uses AJAX. Users of this software can simultaneously modify the same page thanks to AJAX, and any changes are instantly visible to everyone.

AJAX is frequently regarded as more effective and user-friendly than other web development strategies. Anytime an individual interacts with a normal web app, the entire page must be refreshed. Users may find this tedious and slow. Thanks to AJAX, users can now access page updates instantly and without any discernible lag, which enhances the effortlessness and simplicity of their surfing.

AJAX has greatly changed the online development business since it allows programmers to design fluid, active, and adaptive web-based programmes. That is a preferred choice for engineers from a variety of industries because it may enhance the accessibility and performance of web apps.

```
<!-- Add JavaScript code -->
<script>
    $(document).ready(function() {
        $('.ActivationBtn').click(function(event) {
            $.ajax({
                url: "/showUser/getUser?userId="+this.id, success: function(response) {
                    window.location.reload();
                }
            });
        });
    });
</script>
<!-- Initialize DataTable -->
<script>
    $(document).ready(function() {
        $('#usersTable').DataTable({
            "lengthMenu": [ 5, 10, 30 ],
            "pageLength": 20
        });
    });
</script>
</body>
```
> body > script

Fig   3.1:Ajax call for Users

3.4 Groovy

The Java Virtual Machine (JVM) supports the dynamic, object-oriented programming language known as Groovy. It is frequently used to write scripts, create automation tools, and create web applications since it is intended to be succinct, expressive, and versatile. Groovy is a popular option for Java developers because it combines the simplicity of scripting languages like Python and Ruby with the strength and performance of Java.

Java syntax is comparable to Groovy syntax, however Groovy has a number of improvements and simplifications. For instance, Groovy allows closures, which are code snippets that, like anonymous functions in other languages, can be passed around and executed at a later time. Additionally, Groovy comes with built-in support for JSON, collections, and regular expressions, making it simpler to work with these popular data types.

The integration of Groovy with Java is one of its important characteristics. Groovy can directly access Java libraries and frameworks, and Java code

35

can call Groovy code and vice versa. Because of this, integrating Groovy into Java projects that already exist or using Java libraries in Groovy code is simple.

Additionally, Groovy has a number of features that make it a good choice for web development. It features built-in support for generating and consuming web services, and numerous prominent web frameworks, such as Grails and Ratpack, are developed on top of Groovy.

Overall, Groovy is a robust and flexible language that combines the power of Java with the ease of use of scripting languages. Because of how easy it is to use and how well it integrates with Java, it is a popular choice for developers working on a variety of projects.

The following are some advantages of using the programming language Groovy

1) Simple and intuitive syntax makes Groovy simple to learn and use, especially foremost with a background in Java or other related languages.

2) Type verification occurs at runtime rather than during compilation because Groovy is a dynamically typed language. As a result, it is simple to create code quickly and modify it as necessary.

3) Supports both the functional and object-oriented paradigms of programming Groovy supports both the functional and object-oriented paradigms of programming, letting developers select the strategy that best suits their requirements.

4) Integration with Java is seamless because to Groovy's full compatibility with Java and ease of integration with existing Java code, libraries, and frameworks.

5) Rich set of features: Groovy provides a rich set of features, including closures, dynamic typing, operator overloading, and support for both static and dynamic compilation.

36

6) Concise and expressive syntax: Developers can write code in Groovy that is shorter and simpler to comprehend than comparable Java code thanks to the language's concise and expressive grammar.

7) Rapid development: Groovy is well suited for rapid prototyping and development because of its dynamic nature and support for meta-programming, which enable developers to write code more quickly.

8) Strong support for testing: Groovy has strong support for testing, with built-in support over unit evaluation and integrated testing, and can be easily integrated with popular testing frameworks like as JUnit and Spock.

9) Overall, Groovy is a powerful and flexible language that offers many advantages to developers, particularly those working on projects that require a high degree of flexibility and rapid development.

Groovy Collection

Groovy uses collections to organise and manage sets of linked things. Groovy collections are comparable to Java collections, yet they give significant advantages and simplifications.

The following are some essential qualities of Groovy collections:

Groovy collections can be used to handle and store any type of object because they are meant to be strong and adaptable.

1) Improved syntax: Groovy has support for collection literals, range operators, and the spread operator, making it simple to create and use collections.

2) Built-in support for popular data types: Groovy collections give built-in support for standard data types such as lists, maps, and sets.

3) Improved iteration: The each(), eachWithIndex(), and collect() methods are all supported by Groovy's improved iteration capabilities for collections.

4) Groovy collections' support for closures makes it simple for developers to carry out complicated operations on collections.

5) Groovy offers immutable versions of popular collections like lists and maps, which can be helpful in circumstances where data shouldn't be changed.

6) Fluent APIs: Groovy collections provide a fluent API, allowing developers to chain together multiple operations on a collection in a single statement.Thanks to their simplified syntax, enhanced iteration capabilities, and support for closures, they are ideal for a range of tasks, from simple data manipulation to complex data processing.

7) The storage and manipulation of groupings of related objects is done in Groovy via collections. Groovy collections are comparable to Java collections, although they provide various advantages and simplifications.

Listed here are a few of the most popular Groovy collections:

1) Lists: Lists are ordered collections that can contain duplicate elements. By using their position in the list, they can be indexed and accessed.

2) Collections called maps are used to store key-value pairs. A unique key can be used to store and retrieve data using them.

3) Sets: Unordered collections with distinct elements are known as sets. Duplicates in a collection can be eliminated using them.

4) Ranges: A range is a set of values that spans an initial and final value. A run of numbers or characters can be represented by them.

5) Arrays: Arrays are collections with a fixed size that can hold a certain kind of data.

6) Collections called queues keep track of the sequence in which elements were added. They can be used to implement a first-in-first-out (FIFO) data structure.

7) Collections called stacks keep track of the sequence in which their elements were added.

8) Stacks are collections that record the order in which its elements were added. They can be used to implement the LIFO data structure, which stands for last-in, first-out.

9) Groovy collections make working with groups of related objects powerful and flexible. Thanks to their reduced syntax, enhanced iteration capabilities, and support for closures, they are ideal for a variety of tasks, from fundamental data manipulation to sophisticated iteration.

Groovy Closure

Groovy collections make working with groups of related objects powerful and flexible. Thanks to their reduced syntax, enhanced iteration capabilities, and support for closures, they are ideal for a variety of tasks, from fundamental data manipulation to sophisticated iteration.

The Java Virtual Machine (JVM) is the platform on which Groovy, a dynamic language, runs. It was developed by James Strachan in 2003 and is frequently contrasted with Java, a statically typed language that also utilises the JVM. Support for closures is one of the primary characteristics that separates Groovy from Java.

A section of code that can be used as a variable is known as a closure. It can be returned from a method, allocated to a variable, or passed as a parameter to a method. Closures are frequently used to package transferable behaviour within programmes.

Closures have been a part of programming languages for a very long time. Since their initial introduction in Lisp in the late 1950s, they have been implemented into a number of other languages, such as Python, Ruby, and JavaScript. Groovy implements closures in a unique way that has had a significant impact on the programming world.

Groovy support for closures has had a big impact on the programming community. Closures have become more popular in other languages like Java and C# as a result of Groovy success with them.

```
                                                        Live Demo
class Example {
    static void main(String[] args) {
        def clos = {param->println "Hello ${param}"};
        clos.call("World");
    }
}
```

Fig 3.2:Groovy closure Example

Groovy VS Java

It's customary to contrast Java, a statically typed language that also makes use of the Java Virtual Machine (JVM), with Groovy, a dynamic language that runs on the JVM. The following are the primary differences between the two:

1) Groovy's syntax is less convoluted and shorter than Java's. Additionally, it provides a number of features, such as closures and dynamic type, that Java does not.

2) Because Java is a statically typed language, its type system demands that variable types be declared at build time. Variable types are chosen dynamically in Groovy since it has a dynamic typing system.

3) Groovy offers support for metaprogramming, which enables programmers to alter the behaviour of classes while they are running. Using this, it's possible to give current Java classes new capabilities.

4) Writing unit tests and integration tests for apps is made simpler by the

built-in testing facilities provided by Groovy.

5) Performance: Because Java is a compiled language and has a stricter type system than Groovy, it runs faster in general. Groovy's dynamic nature, on the other hand, can make the language more adaptable and make writing rapid code easier.

6) The choice between Java and Groovy for a project will primarily depend on its requirements and the preferences of the development team. Both languages have benefits and drawbacks overall.

3.5 Hibernate

A Java-based object-relational mapping (ORM) framework called Hibernate offers a high-level, object-oriented API for interacting with relational databases and lets developers map Java objects to database tables. It was developed to reduce the need for manually writing JDBC (Java Database Connectivity) code while developing Java applications that use databases.

Hibernate is an open-source technology that is commonly used by enterprise Java applications. It supports several databases, including MySQL, Oracle, PostgreSQL, and SQL Server, and offers a reliable and adaptable method for working with databases.

Hibernate's ability to abstract away much of the low-level JDBC code needed to communicate with databases is one of its key advantages. Instead, a high-level, object-oriented API is available for developers to use in order to interface with the database, speeding up, streamlining, and reducing the likelihood of mistakes.Hibernate also provides a variety of other capabilities and benefits, including:

1) Support for caching and lazy loading: Hibernate provides a powerful caching mechanism that can help to improve application performance by reducing the number of database queries required. Additionally, it supports lazy loading, which means that rather than loading all data at once, data is only loaded from the database when it is actually required.

2) HQL (Hibernate Query Language), a robust object-oriented query language provided by Hibernate, enables programmers to write queries in terms of Java objects rather than SQL tables and columns.

3) Hibernate offers support for transactions, enabling programmers to combine many database operations into a single, atomic unit of work.

4) Integration with other frameworks: Spring and Struts are two notable Java frameworks that Hibernate can be readily connected with.

5) All things considered, Hibernate is a strong and adaptable ORM framework that may assist programmers in creating database-driven applications more rapidly, effectively, and with less prone to errors.Instead of loading all data at once, data is only pulled from the database when it is actually needed.

Hibernate Benefits

Object-Relational Mapping (ORM) framework for Java known as Hibernate offers a number of advantages, such as:

1) Simplified database access: Hibernate reduces the amount of code needed for database operations by abstracting away the low-level complexities of working with a database.

2) Hibernate's object-oriented design allows developers to work with Java objects rather than SQL statements, enabling a more natural and simple way to interact with data.

3) Enhanced productivity: By eliminating the need for manual SQL writing, Hibernate's automatic mapping of Java classes to database tables can help developers save time and effort.

4) It is feasible to create applications with a single codebase that can be utilised in a variety of circumstances thanks to Hibernate's support for several database platforms.

5) Performance: Hibernate provides efficient caching techniques as well as other performance improvements that can help speed up database operations.

6) Data consistency: Hibernates transaction management service ensures that database operations are carried out consistently and reliably.

7) Scalability and maintenance: Because to Hibernate, database applications are simpler to scale and maintain because the framework clearly distinguishes between code for the application and code for the database.

Linux

Based on the Unix operating system, Linux is a free and open-source operating system. Since its creation by Linus Torvalds in 1991, it has grown to be among the most widely used operating systems worldwide, especially in the server and embedded systems sectors. Linux is widely used by developers, companies, and people worldwide and is renowned for its reliability, security, and flexibility.

With a variety of distributions readily available that are tailored for particular use cases, including servers, desktops, and embedded computers, Linux is also renowned for its adaptability. The most well-known Linux distributions include CentOS, Fedora, Ubuntu, and Debian.

The security of Linux is another important aspect. Due in part to its open-source nature and developers' ability to swiftly fix security flaws, Linux has a reputation for being more safe than other operating systems. Additionally, Linux contains a number of built-in security mechanisms that assist in preventing unauthorized access to sensitive data, such as file permissions and access controls.

Last but not least, Linux is renowned for its adaptability. A variety of software, including programming languages, development tools, and web servers, is accessible for installation. Because of this, it makes for the perfect platform for developers and companies looking for a versatile and customized operating system.

Generally speaking, Linux is a strong and adaptable operating system that provides a number of advantages, such as stability, security, and flexibility. It is widely used by companies, developers, and people all around the world

and has a sizable and vibrant developer community.

Using Linux as an operating system has a lot of benefits. Some of the main benefits are as follows:

Cost: Linux is free and open-source, therefore using it does not need purchasing a licence. Particularly for enterprises that must install the operating system on numerous machines, this might result in significant cost savings.

Flexibility: Linux offers a wide choice of distributions that may be customised to meet a variety of requirements. It is therefore the perfect platform for programmers and companies that want a versatile and adaptable operating system.

Security: Linux has a reputation for being secure, and it has a number of built-in security mechanisms, such file permissions and access controls, to help prevent unauthorised access to sensitive data. Additionally, as it is open-source, programmers can immediately repair security holes as they are found.

Stability: Linux has a reputation for being able to operate for extended periods of time without crashing or needing to be restarted. Businesses that need to operate crucial applications continuously may find this to be of special importance.

Performance: Linux is renowned for its great performance, and it can run on outdated hardware while still delivering quick performance on even entry-level computers. This makes it the perfect platform for companies and people who want to make the most of their hardware.

Community: The Linux operating system has a sizable and vibrant community of users and developers who are continually striving to enhance the system and support one another. For companies and people looking for assistance or advice on using Linux, this community can be a useful

resource.

All things considered, Linux is a strong and adaptable operating system that provides a number of advantages, including cost savings, customizability, security, stability, performance, and community support. Businesses, developers, and people all over the world use it extensively.

3.6 GSP

GSP (Groovy Server Pages) is the default view technology for the Grails framework. GSP enables developers to create dynamic web pages that are easily produced from the controller using the model-view-controller (MVC) paradigm.

By combining HTML and Groovy code to generate GSP sites, developers can take use of the language's flexibility and power while still adhering to approved web development standards. Manage user interactions, check user input, and show data from a controller using GSP pages.

One of the key features of GSP is the use of tag libraries, which provide an easy way to add complex functionality to a web page without having to write custom code. Use the built-in tag libraries that come with Grails, such as the form tag library, to create HTML forms that are automatically filled with data from the controller.

GSP pages can be readily modified using CSS and JavaScript by developers to create complex, interactive user interfaces. By defining a uniform layout for many pages using the layout templates provided by GSP, the amount of duplicate code that needs to be written is reduced.

All things considered, GSP is a reliable and flexible view technology that provides developers with a wide range of tools and features for creating dynamic web pages within the Grails framework. It is well-liked by Grails users and effective for building complex, data-driven web applications.

```
<!-- Profile Dropdown -->
<li class="nav-item dropdown mx-4">

    <ul class="dropdown-menu dropdown-menu-end" aria-labelledby="navbarDropdown">
        <li>
            <g:link controller="profile" action="index" class="dropdown-item">Profile</g:link>
        </li>
        <li><a class="dropdown-item" href="#">Users</a></li>
        <li><a class="dropdown-item" href="#">Topics</a></li>
        <li><hr class="dropdown-divider"/></li>
        <li><a class="dropdown-item" href="#">Posts</a></li>
        <li>
            <g:link controller="logout" action="index" class="dropdown-item">Logout</g:link>
        </li>
    </ul>
</li>
<li class="pt-3">
    ${user.username}
</li>
<li class="nav-item dropdown mx-4">
```

Fig. 3.3 : GSP Form

GSP Features

A robust and well-liked JavaScript programme called jQuery makes it easier to use the Document Object Model (DOM) to create dynamic, interactive web pages. The key characteristics of jQuery include stuff like:

DOM Manipulation: Thanks to jQuery's straightforward and user-friendly syntax for DOM manipulation, developers may quickly add, remove, and edit items on a web page.
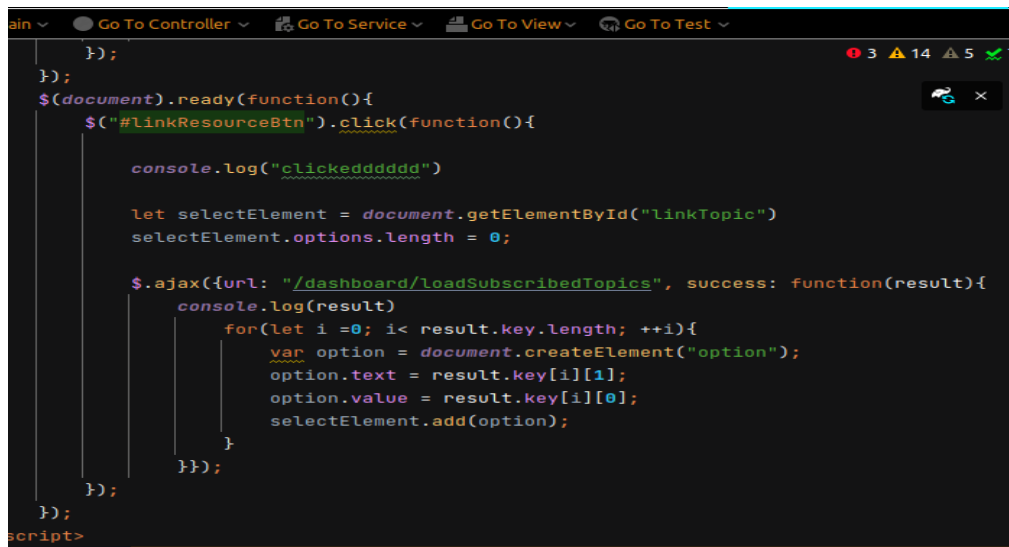
Event Handling: With jQuery, programmers can quickly handle user events like keystrokes, mouse clicks, and form submissions, enabling them to design responsive and engaging user interfaces.

AJAX: JQuery makes it easier to create dynamic online applications by allowing developers to load content from a server without refreshing the page.

Effects and Animations: Making interesting user interfaces that respond to user input is made simpler by using jQuery's pre-built animations and effects, such as slide-outs and fade-ins.

Cross-Browser Compatibility: By addressing the variations in browser

46

implementations.

Fig 3.4 AJAX Use Case

Extensibility: A variety of jQuery plugins are available that add new functions including data validation, form administration, and complex animations.

Simplified Syntax: jQuery's syntax is easier for developers to grasp and write than pure JavaScript, allowing them to construct more robust online applications with less code.

Overall, jQuery is a reliable and flexible JavaScript framework that speeds up online development and makes it easier to create dynamic, interactive, and cross-browser compatible websites. Every web developer should have it as it is a necessary tool that is frequently used in the industry.

JDBC

The JDBC standard API (Application Programming Interface) enables communication between Java programmes and relational databases. Grails uses JDBC to interact with relational databases like MySQL, PostgreSQL, Oracle, and others.

The JDBC API for Grails speeds up database access by offering a higher-

47

level API than standard JDBC. An efficient interface for managing transactions, running SQL statements, and transforming query results to objects is provided via the Grails JDBC API.

You must first specify the specifics of the database connection in the DataSource.groovy file before you can use JDBC in Grails. Usually, you can find this file under the grails-app/conf/ directory.

Users of Grails now have a reliable and flexible way to communicate with relational databases thanks to JDBC. It is easier to work with JDBC because of the Grails JDBC API's streamlined API and handling of many of the low-level details of database access.

JDBC Features

The Java API known as JDBC (Java Database Connectivity) offers a standardised

method for Java programmes to communicate with relational databases. Among

JDBC's primary characteristics are:

1) Connectivity to Relational Databases: JDBC offers a standardised method for

connecting to a variety of relational databases, including Oracle, MySQL,

PostgreSQL, Microsoft SQL Server, etc.

2) Platform Independence: JDBC offers a method of interacting with databases

that is independent of the platform being used, making it simple to create

database applications that can operate on any system that supports Java.

3) Support for SQL: JDBC offers a comprehensive collection of classes and interfaces

for interacting with databases using SQL (Structured Query Language), the

common language for interacting with relational databases.

4) JDBC supports transactions, which let you combine several database operations

into a single transaction and guarantee that either all of the operations in the transaction

are properly completed or none of them are.

5)Metadata Access: JDBC makes it simple to create general database applications

that can be used with any database by providing access to database metadata, such as

table and column names, data types, indexes, and constraints.

Javascript

Web development usually uses high-level, interpreted programming dialects like JavaScript. It was created by Netscape and released for use in 1995. It is

presently among the programming tongues that are most often used globally.

Imperative, functional, and object-oriented programming paradigms are all supported

by JavaScript, an evolving programming language. It is widely coupled with HTML

and CSS to create interactive websites and applications for the web.

Listed below are some of JavaScript's key attributes:

1) Using object-oriented programming JavaScript is an object-oriented language built on prototypes. This indicates that without the usage of constructors or classes, things can be constructed quickly.

2) Acts as a first-class item: In JavaScript, functions are first-class objects. This suggests that they can be used as functions' return values, passed around as values, and given to variables.

3) Because JavaScript utilities dynamic typing, the types of variables are chosen while the programme is being run. This facilitates quick code development, but if you're not attentive, it could also lead to mistakes.

4) JavaScript provides robust asynchronous programming features, enabling non-blocking I/O operations that can improve the speed of web applications.

5) Client-side scripting, which enables dynamic interactive web pages and user interfaces, is typically done using JavaScript.

In general, JavaScript is a reliable and flexible language that has turned into a crucial tool for web developers. It is used for a variety of purposes, such as server-side development using tools like Node.js for everything from simple websites to sophisticated online software.

Javascript Features

Here are some key features of JavaScript:

1) Object-oriented programming: JavaScript is a prototype-based object-oriented language. This means that objects can be created on the fly, without the need for classes or constructors.

2) JavaScript is an object-oriented language with a prototype-based grammar. This shows that things can be swiftly produced without the need of constructors or classes.

3) Because JavaScript utilises dynamic typing, the types of variables are chosen while the programme is being run. This facilitates quick code development, but if you're not attentive, it could also lead to mistakes.

4) JavaScript provides robust asynchronous programming features, enabling non-blocking I/O operations that can improve the speed of web applications.

5) Client-side scripting, which enables dynamic interactive web pages and user interfaces, is typically done using JavaScript.

6) User-event-responsive programming: JavaScript can respond to user activities like clicks, scrolls, and input because it is an event-driven language.

7) Cross-platform compatibility: Any device with a web browser can use JavaScript, an extremely flexible language.

8) Simple to learn: Learning JavaScript is said to be rather simple for individuals who are already familiar with HTML and CSS.

9) Large developer community and resources: Due to the size and activity of the development of JavaScript belonging, their are many learning and troubleshooting resources available.

In general, JavaScript is a powerful and flexible language that is increasingly necessary for developing modern web applications.

CHAPTER- 4 RESULTS AND ANALYSIS

Response time is the amount of time it takes for the system to respond to user queries. Response time is measured in this project by the time it takes for the software to reload the consumer's a dashboard and display a listing of registered topics, and present the resources associated with each topic. This programme should respond in less than 2 seconds.

Throughput is the number of queries completed by the system's components per unit time. Good throughput for this application would be the ability to accommodate multiple users concurrently without any delays or difficulties.

The maximum amount of concurrent users that can use the system as well as the total amount of responses completed by the system within a given time period can be used to calculate the project's throughput.

The ability of a system to handle a growing number of consumers or requests while maintaining performance is referred to as scalability. The number of users that may join the database simultaneously as well as the range of topics and material that will be added to the system, are indicators of scalability in this project. An effective scalable system can handle an immense amount of users as well as information without reducing performance significantly.
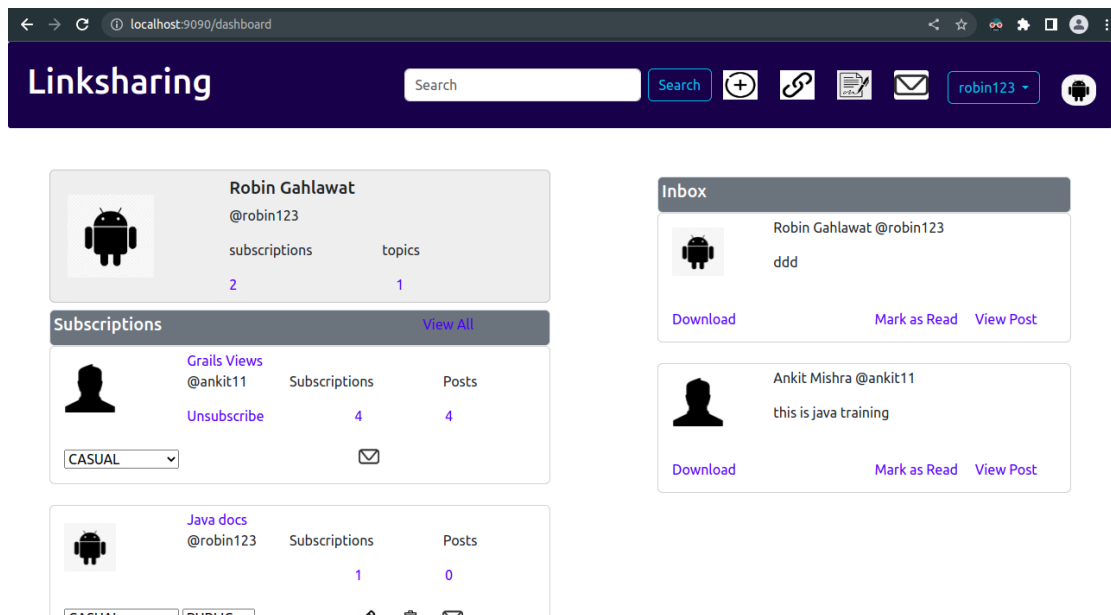
Fig 4.1 Linksharing Dashboard

Availability refers to a system's ability to stay working and readily available to users irrespective of whether hardware or software fails. In the case of this project, availability can be measured via the measure of the system's uptime, or the quantity of time that the system is working and available to users. The availability of this application ought to be a minimum of 99.9%.

Several strategies, which include caches, distributing the load, and database optimisation, can be used to ensure satisfactory performance in the aforementioned project. Warehousing can be used to keep data that is frequently used in memory, lowering system response time. Load balancing is a technique for distributing the load among different servers and guaranteeing that the system can manage an excessive amount of users and requests. Database optimisation is capable of helping ensure that database queries are optimised for performance, hence decreasing the time required to obtain data from the database.
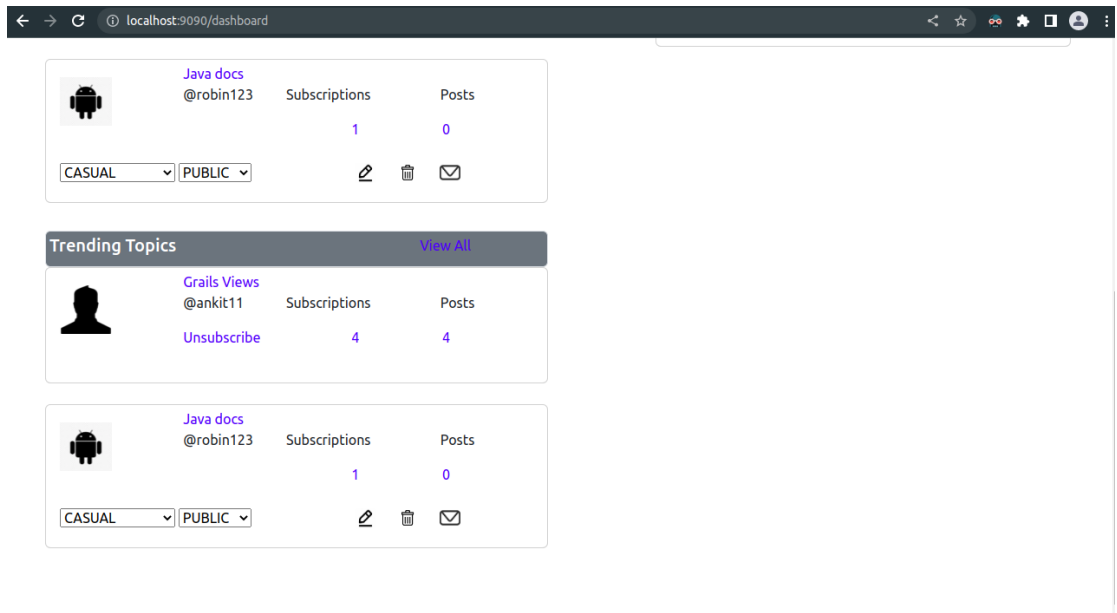
Fig 4.2 Trending Topics

CHAPTER -5

CONCLUSION

5.1 Conclusion

The process of creating the application was a useful learning opportunity for creating web applications with Grails. The development process was quicker and more effective when the Grails framework and Groovy language were used instead of conventional Java frameworks. The usage of GSP technology also simplified the process of developing dynamic web pages and decreased the amount of code needed to generate the views.

The platform of the programme allows users to create, subscribe to, and add resources to subjects. The administration user's presence enables content control and ensures that unwanted content can be removed. Application of secure coding methods and security procedures ensures the safety of users' data.

The controllers and views for the application were constructed once the database schema was created. The controllers were in charge of receiving user requests and calling the proper methods in the service layer to carry out the necessary operations. The views were in charge of presenting the data to the viewers as HTML pages. The Groovy Server Pages (GSP) technology, which enables programmers to generate dynamic web pages using the Groovy language, was used to create the views.

In conclusion, the project met its goals and offered a useful learning opportunity for creating web applications using Grails and the Groovy programming language.

At last, the advancement interaction was enhanced involving GitHub Activities for computerized testing, code inclusion, and linter tests. In general, the Linksharing Programming interface was established

utilizing a purview of best practices and strategies, bringing about a top notch, dependable, and performant Programminginterface that addressed the issues of its clients.

5.2 Goals achieved

The Linksharing web applicatoin that we created using Microservices architecture has achieved almost all of our required goals that we setup in starting of the project.

Versatility: The three-layer engineering, utilization of microservices, and execution of Kubernetes deemed the Programming interface to effectively scale and manage expanded traffic as the application developed.

Unwavering quality: The execution of TDD, combination testing, and continuous coordination/constant arrangement (CI/Album) through GitHub Activities guaranteed that the Programming interface was dependable and liberated from defects.

Security: Best practices for verification and approval, input approval, and Programming interface key administration were carried out to guarantee the security of the Programming interface.

Practicality: The utilization of irrefutably factual code, data set relocations, and organized recording made the Programming interface simple to keep up with and update on a case by case basis.

Execution: The execution of metrics observing, execution investigation, and enhancements worked on the Programming interface's general exhibition and reactiontime.

In general, the Linksharing Programming interface accomplished its objectives of providing a versatile, dependable, secure, viable, and performant resource sharing arrangement.

5.3 Future Scope:

There are a few areas of future degree for the ZopStore Programming interface:

Adding new elements: The ZopStore Programming interface can be

reached out with new highlights to make it more valuable for organizations and purchasers. For instance, adding support for numerous installment entryways, item audits, and client appraisals could improve the client experience and augment client commitment.

Reconciliation with different frameworks: The Programming interface can be incorporated with different frameworks like ERP, CRM, and coordinated factors of the administrators to give a consistent start to finish answer for organizations.

Improving security: Similarly as with any web application, security is generally a concern. The Programming interface can be additionally enhanced to incorporate extra security elements like two-factor verification, encryption, and interruption identification.

Enhancement: The Programming interface can be advanced for speedier reaction times and further developed versatility. This can be accomplished through strategies, for example, load testing, execution profiling, and code enhancement.

Supporting multiple stages: While the ZopStore Programming interface was intended to be utilized with Golang, it tends to be extended to help other programming dialects like Python or Node.js. This could expand its reception and make it more open to a more extensive scope of engineers.

Cloud arrangement: The Programming interface can be communicated on cloud stages like AWS, Sky blue, or Google Cloud, which would offer extra advantages like adaptability, unwavering quality, and cost-adequacy.

Generally speaking, there are a few energizing open doors for future turn of events and development of the Linksharing Programming interface, and it will be fascinating to discern how it develops over the long term.

References

1 https://groovy-lang.org/documentation.html [Online]

2 https://docs.grails.org/3.3.9/guide/index.html [Online]

3 https://www.w3schools.com/js/   [Online]

4 https://www.w3schools.com/jquery/   [Online]

5 https://www.w3schools.com/js/js_ajax_intro.asp/   [Online]

6 https://www.w3schools.com/html/   [Online].

7 https://www.w3schools.com/css/   [Online].

8 https://getbootstrap.com/docs/4.1/components/alerts

Appendix

Appendix A: Technologies Used

☐ Programming language: Groovy

☐ Framework: Grails

☐ Database: Oracle

☐ API documentation: Grails Documentation

☐ Approach: Micro Services

☐ Mocking framework: Groovy on Grails

Appendix B: API Endpoints

Endpoint	HTTP Method	Description
/login	GET	Open the login page
/register	GET	Open the register page
/profile	GET	Open the profile page
/dashboard	GET	Open the dashboard
/topic	GET	Delete a specific customer by ID
/resource	GET	Add a new vehicle

A Mishra

10	eprints.utem.edu.my Internet Source	<1%
11	Submitted to Dhirubhai Ambani Institute of Information and Communication Student Paper	<1%
12	repository.sustech.edu Internet Source	<1%
13	Submitted to University of Westminster Student Paper	<1%
14	dspace.uowm.gr Internet Source	<1%
15	www.seofever.com Internet Source	<1%
16	www.udemy.com Internet Source	<1%
17	dspace.cvut.cz Internet Source	<1%
18	erepository.uonbi.ac.ke Internet Source	<1%
19	vtechworks.lib.vt.edu Internet Source	<1%
20	www.slideshare.net Internet Source	<1%
21	brage.bibsys.no Internet Source	