HOCHSCHULE SRH HEIDELBERG

MODULE TECHNICAL DESIGN

Labor and Employment System

Geo Joseph(11011939)	Praveen Kumar Appari(11012160)
Kushal pulluru(11011787)	Sreehari Mullapulli(11011996)
Niveditha Bangalore Ramanujan(11011826)	veerendhar Deevanapalli(11010931)

Creation Date : June, 03, 2019

Last Updated : June, 14, 2019

Document Ref : Sample of university provided

Version : 1.3

Approvals:

Pro. Dr. Ajinkya Prabhune

Pro. Dr. Barbara Sprick

Document Control

Change Record

Date	Author	Version	Change Reference
24.05.2019		1.1	Design user stories
29.05.2019		1.2	Changes in user stories and data model
03.06.2019		1.3	Change of one user story

Reviewers

Name	Subject
Pro. Dr. Ajinkya Prabhune	Information Systems
Pro. Dr. Barbara Sprick	Information Systems

Contents

1. Introduction

The investigation of Legal Systems suggests to a methodology or procedure for translating and upholding the laws. It explains the rights and duties in an assortment of ways. There is always a changes and updates in act of laws published in the database. This information is always required and used for research, information and general interest and rights to know the law details.

The purpose of providing this Labor and Employment laws information in form of discussion forum as to have Democratic exchange. Organizing a data with huge information and transaction is difficult. So this system has been implemented using Redis, MongoDB and Neo4j which are NoSQL databases. MongoDB is a object-oriented and scalable NoSQL database which we store the collective informatic data in document format. Redis, is a key-value database was used to create the cache and ranking of the threads. Neo4j, graph database is used to map the relationships.

The discussion forum enables users to have a platform to discuss about the queries regarding the labor and employment system. Users will have the option to post a thread in the forum and also it enables the users to comment on the post which is posted on the forum.

2.Organization

The project in total have six user stories. The user stories are divided one per person. Each member worked in their individual user stories which were assigned to them. In addition to that each team member also had the opportunity to work with each database, study and get familiarized. The detailed description of meeting schedule and actions along with roles and responsibility are described below.

2.1 Meeting Schedules and Actions

Days	Actions	Attendance	Duration
14 May 2019 to 16 May 2019	User Case Framing Details: Each member is assigned to come up with ideas about relationship between people's life and democracy. Different ideas were discussed during meeting and finalized the root subject as labor and employment system	Geo Kusal Niveditha Praveen Sreehari Veerendhar	Each day between 2 to 4 hours
17 May 2019 to 24 May 2019	User Stories Details: User stories were framed up by each member which is then discussed in the meeting. Corrections were made up on different user story. Exchange of ideas for user story framing	Geo Kusal Niveditha Praveen Sreehari Veerendhar	Each day between 2 to 4 hours
25 May 2019 to 29 May 2019	Tools and User Stories Details: Discussed about the tools to be used for the database design along with assigning each member to ideate the user story.	Geo Kusal Niveditha Praveen Sreehari Veerendhar	Each day between 2 to 4 hours

Days	Actions	Attendance	Duration
30 May 2019 to 03 June 2019	User Story, Data Model Details: As discussed with the Professors some user stories are changed. The data model design along with data collection for user story implementation.	Geo Kusal Niveditha Praveen Sreehari Veerendhar	Each day between 2 to 4 hours
04 June 2019 to 11 June 2019	Database Implementation Details: Getting familiarized with the databases. The user story implementation in mongo dB, Redis and neo4j were done. Team member assigned for particular user story discussed his/her discovering about the idea and implementation in DB layer. Alpha version is made available on 11 June 2019.	Geo Kusal Niveditha Praveen Sreehari Veerendhar	Each day between 2 to 4 hours
11 June 2019 to 14 June 2019	Database Review with Modifications Details: Design along with the queries had some modifications. The final exam presentations were made. Had a trial for the final presentation.	Geo Kusal Niveditha Praveen Sreehari Veerendhar	Each day between 2 to 4 hours
14 June 2019 to 16 June 2019	Documentation Details: The Final report was made.	Geo Kusal Praveen Sreehari Veerendhar	Each day between 2 to 4 hours

2.2 Tools Used:

The meetings were organized through WhatsApp. Content and activity updates are done using Trello. Resources are shared using the google drive. For Neo4j database we made use of http://www.apcjones.com/arrows/.

1.WhatsApp

2.Trello

3.Google Drive

4. http://www.apcjones.com/arrows/.

5. https://www.draw.io/

2.3 Tasks Role and Responsibility

Responsible= R

Accountable=A

Consultant=C

Activities	Geo Joseph	Kushal	Niveditha	Praveen Kumar	Sreehari	Veerendhar
Requirement gathering and Scope analysis	R	R	R	R	R	R
Brain storming	R	R	R	R	R	R
Defining Use case and User stories	R	R	R	R	R	R
Moscow Analysis	R	R	R	R	R	R
Data Modeling	R	R	R	R	R	R
Data collection	R	R	R	R	R	R
Design and Development						
MongoDB	R	A	A	R	C	R
Neo4j	A	R	C	C	A	C
Redis	C	C	R	A	R	A
Documentation	R	A	C	R	A	A

3. Project Use Case

Under reference of format provided by federal minister of justice and consumer protection with all basic federal law for the Federal Republic of Germany. Labor and Employment Laws is one of the strongest pillars to form the standards which covers common issues in employment and labor laws and regulations – terms and conditions of the employment, employee representation and industrial relations, discrimination, maternity and family leave rights and business sales within 51 - jurisdictions. These laws have stabilized with the information set of rules to the people to follow, educate. Considering the all international and national issues to have proper control over labor and employment laws these acts of laws had been embedded by the German government.

Considering in our project use case, goal is to implement the data model for Discussion Form for existed labor and employment laws with their user queries. In this schema it should provide a discussion for users to post the article regarding employment laws and users have flexibility to comment the post with their informative comments, attain knowledge, vote the post which is posted by the user. Collective information regarding employment has been stored. Referring with laws users post their articles with their questionnaires. Here we maintain two users as Admin and common user. This model is designed for educating with the discussion in application and by providing the data to users and develop the core factor of the application too.

User Story: 1

"As an end user, I would like to know about the legal working hours and relevant discussions in discussion forum so that I will be aware of the information"

The end-user of the discussion would like to retrieve relevant laws of legal working hours of Germany and corresponding threads and comments for that legal laws according to his choice.

Requirements:

Laws to be dumped as document by admin
Threads and comments created for legal working related laws by end user.
Users can able to comment on the article available.
Common user can provide the upvoting and down voting to the article posted by end-user.

Aggregating the laws related with article with user comments provided and vice-versa.

User Story: 2

"As an admin of the forum. I wants to update the content of minimum wage policy since there is a change to the law. "

The Admin of the application is liable to update the laws that has been provided with user comments and relevant confirmations as well as proof. In database what ever the laws have been updated then the previous data must be stored with the part of version control and pointing predecessor and successor log.

Requirements:

Admin should have rights to manipulate the data where should be authorized to insert/update the law with relevant information by user comments. Reference
Old and new laws must be linked so that there should be backup of changes done in dump

User Story: 3

"As an end user, I would like to get users opinion on his research topic about temporary employment law by initiating a discussion in the discussion forum, so that I can improvise my research."

The application user would like to know the opinions on the research topic about temporary employment. The user will create post related to the laws. In this article, users can provide their comments which can provide information for user's research.

Requirements:

Legal temporary employment laws must be inserted for the reference of laws by end user and common users.
Users should have created according to 3 categories i.e., Admin, end-user and common user (who are able to provide their updates on the article.)

User Story: 4

"As an active end user of the discussion forum. I wants to get suggestions according to people whom I follow in the forum, so that I will be aware of new topics. "

The user wants to know how feeds are suggested to end user on which context and how intelligent, real-time persisting relationships and connections through every transition of existence that tackle today's toughest enterprise challenges.

Requirements:

In order to achieve this, we should reveal invisible contexts and hidden relationships by intuitively points and the connections between nodes.
Options for storing connected data to implement a data model, to operate using a query language and to persistence within a sackable, reliable database.
Building a graph database.
Graph-based suggestions are capabilities to deliver only relevant results as feeds to end users.

User Story: 5

"As an end user, I want to see the top contributors of the forum."

The end user of the application wants to see the top contributors of the portal. Users are ranked according to their contributions in portal like Creating a discussion, Commenting, upvoting and downvoting.

Requirements:

User need to have his account information in database.
Users score board needs to be maintained and updated dynamically in the database and retrieved upon request (the user provides the range and sorting method).

User Story: 6

"As an end user who is using the discussion forum, wants to see the recent and relevant discussions on a particular legal law. This would provide me with a chance to get familiarized with recent information and users' opinions about that law."

The end user is in a legal section page and he/she wants to see the most relevant discussions on that section and corresponding comments for that discussion.

Requirements:

User navigated to a legal section page.
A score board for each legal section and all threads under that ranked real-time and maintained according the comments, upvotes and downvotes maintained in database.
Threads and comments information also should be maintained in database.

4. Fundamentals

4.1Technology Used

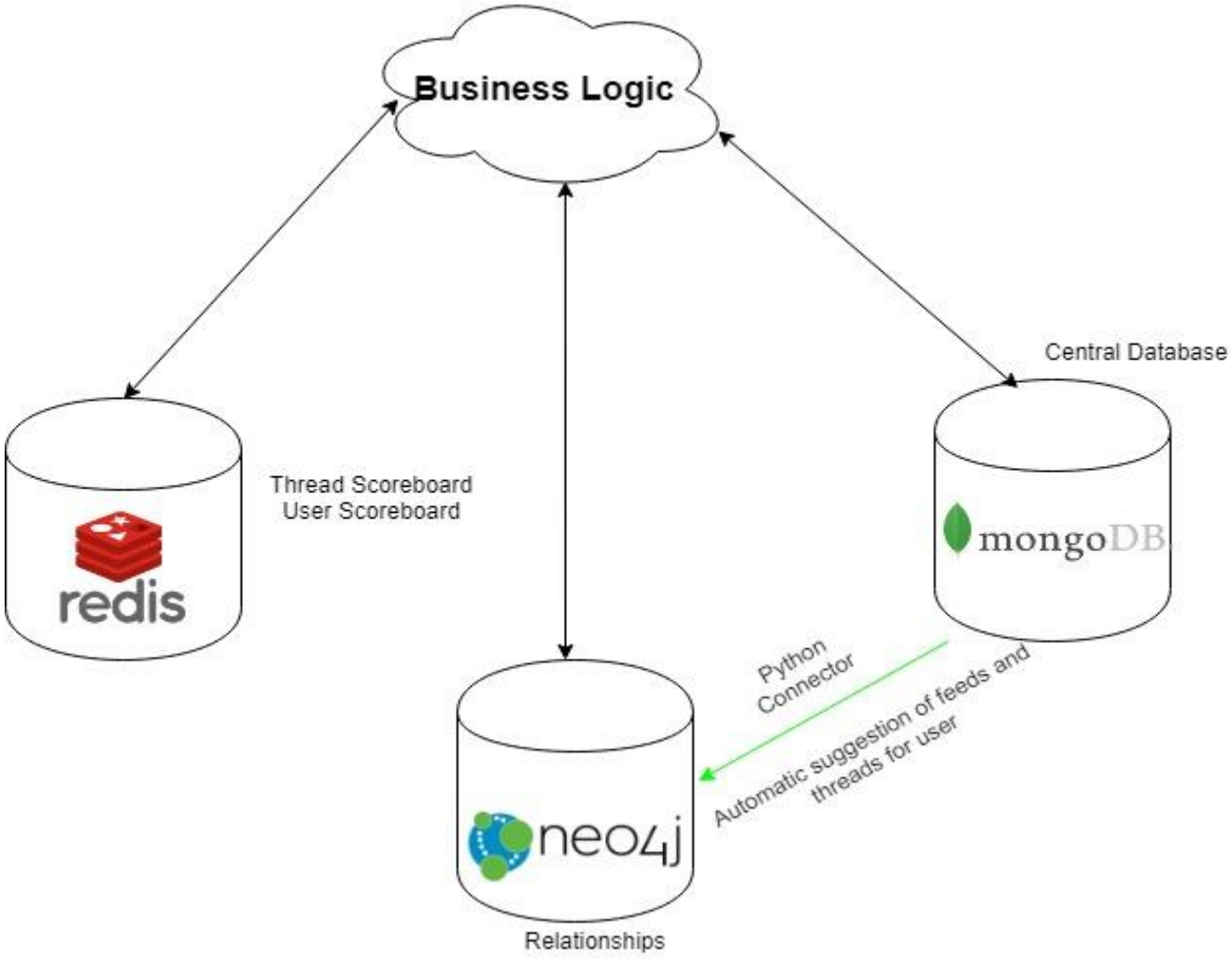
A database is a set of related information which are organized together. The user stories are implemented using NoSQL databases. For the implementation of discussion forum - MongoDB, Redis and Neo4j were used which are the key value, document oriented, and graph databases respectively.

1.Key Value Database: As the name says a key-value store saves data in a key-value data structure where the key is usually a string and the data to save is the value. This type of database ensures the better performance. In the discussion forum key value database is used to keep scoreboard and top contributor.

2. Document Oriented Database: In this project all the detailed information was stored in the document database which stores data in documents. A Document contains one or more fields, each field is a typed value. The typed values can be string, date, binary, array, etc. We have made use of the document database to frame database collections for the discussion forum.

3.Graph Database: A graph database is a set of vertices and edges. It represents entities as nodes and the way these entities relate to each other in the real world as relationships. In this project we made use of Neo4j to map the relationships in discussion forum.

Data Model **fig 4.2**



The design theme of the project is to implement a discussion forum. The users can post threads and can get comments for that. MongoDB is the central data store which contains all the data collections for law, threads and users. Redis will manage the scoreboard for thread and contributor which is used to get the top contributor of the forum with the help of upvotes. Neo4j is used to map the relationships between nodes. Also using a python connector, we are implementing automatic creation, updating of nodes and relationships.

4.3 CRUD Operations

CRUD is an acronym for the four basic types of operations: Create, Read, Update, Delete.

CREATE:

MongoDB

The create queries in MongoDB are used to make a post or thread and comments in the forum.

Neo4j

In Neo4j create operations are used to create nodes and relations.

CREATE (`p` : legal {section_id:'44g01b172807j0535e19d9b7d',section:'Sub_Section_18a'})

Redis

In Redis create operations are used to make the scoreboard.

READ:

MongoDB

The retrieval of the law inserted in the MongoDB is done using the read operation in MongoDB.

db.LegalLaws.find(){"id":ObjectId("5d01b17289509535e09592d6"),"Law_Subsection":"AUG","Law_Subject":"Law_Subject","Law_Subsection":"1","LawSubsection Subject":"Temporary employment,license requirement","LawSubsection":"1a","LawSubsectionSubject":"Display of the transfer"}.pretty()

Neo4j

In Neo4j the read operations will return the graph according to the according to the conditions specified.

MATCH (`p` : legal {section_id:'44g01b172807j0535e19d9b7d',section:'Sub_Section_18a'})
RETURN p

Redis

In Redis using a business model the information will be fetched from the MongoDB.

UPDATE:

MongoDB

In MongoDB the update option is used to update the fields inside the document which is the information regarding laws

Neo4j

The update queries are used to update the specific properties and values of node

```
MATCH (`p` : legal {sec-
tion_id:'44g01b172807j0535e19d9b7d',section:'Sub_Section_18a'})
set  p.createdate = date('2019-05-05')
RETURN p
```

Redis

The scoreboard is getting updated through the Redis command.

DELETE:

In MongoDB, Redis, Neo4j delete operation can be performed but in this data model we are not directly implementing the delete operation as we are maintaining only a change log .

Neo4j

```
MATCH
(`p`:legal{section_id:'44g01b172807j0535e19d9b7d',section:'Sub_Section_18B'})-[rel:
refering]-
>(k:legal{section_id:'54g01b172807j053pe19du369pu',section:'Sub_Section_18c'})
DETACH DELET P

MATCH
(`p`:legal{section_id:'44g01b172807j0535e19d9b7d',section:'Sub_Section_18B'})
DELETE p
```

5. Project Data Model

This project model revolves in discussion form with all legal laws as one entity. The relations to each entity have been scaled below with appropriate design according to model of the project.

MongoDB

The gathered information related to the legal laws are stored in MongoDB as it stored the data in document format and easier to preserve the collective information. This makes the work more efficient in querying the relative document. Hence, we here store these data as collections i.e., legalLaws, users and threads. We have these collections even for discussion form in order to store the data of our application used by the users (See Appendix A)

LegalLaws collection show the detail storage format of how legal laws are stored in document format.

Users collection shows how three types of users are categorized and registered in document format.

Threads collection shows how created article is stored and comments are followed in and as object.

Neo4j

Neo4j data model is designed in such a way to answer questions in the form of Cypher queries and solve business and technical problems by organizing a data structure for the graph database. The gathered information related to Legal laws of Germany are taken in as arbitrary domain as a connected graph of nodes and relationships with properties and labels. Neo4j data model determine the logic of queries and structure data in storage, adapt the changes easily with business logic. Cypher is a declarative graph query language that allows for expressive and efficient querying and updating of the graph. Query performance optimization is to ensure that only necessary data is retrieved from the graph. User will receive new feeds as suggestions from the people he follow by sorting the follow list and results are stored in cache query and hopped next to find the actual section as suggestions through cypher queries and the relation.

Relationships:

Relationships organize nodes into structures, allowing a graph to resemble to richly inter-connected structures. Relationship connects two nodes. There are two types of relations user her : Directed Relationships and Bidirectional Relationships.

NODES:

Nodes are often used to represent entities. Legal nodes are actual data reflection of laws, sections, sub-section present in MongoDB data base. Threads comes under le-gal nodes where user initiates a discussion on section with relevant information. Comment node stores the user unique identification along with comment content. User node stores user basic information as property.

Redis

The legal laws are less transactional, and the threads and comments are heavy transactional type of data. That is the updating, creation and deletion rate happening for threads and comments will be huge. The user wants to see the relevant discussions first under a legal section rather than seeing the less relevant content at the top and scrolling through all the discussions. In order achieve this we have created a scoreboard for each legal law and ranked corresponding threads under

that according to their relevance. Redis is incredibly fast in updating and retrieving the score and scoreboard correspondingly. The sorted data set is used for this purpose.

Engaging and motivating users in the discussion forum is very important for the business. For this purpose, we have created a user score board the same way as legal law scoreboard. Each user is ranked in this score board according to their contributions. The sorted data type only used for this purpose as well.

6.Implementation

6.1 Implementation of Data Model

MongoDB

The Database is created under which collections are created with multiple documents and required fields. LegalLaws collections are created in formal descriptive storage in document according to legal act laws and sub sections and every user detail are stored in user collections, where user have access to perform tasks like comments, read threads, upvote, downvote and create threads. Threads collections are created when user like to create thread, comment on thread posted, upvote and downvote on threads posted. Users and legal laws object id are referencing in threads collection.

Neo4j

After insertion of a section in MongoDB, the Python script triggers the connection to Neo4j only the properties that are needed. Any change in mongo reflects on neo4j graph. Every node should be maintained at-least with one unique property. When its particulars to create a relation between two nodes. Its mandatory to match the node before creating the relation. If not, the relationship is created between two new nodes with no properties. As we mentioned CURD operations in neo4j above with example plays a major role in creating the graph model. Apart from CURD operations we can find as:

Total number of relationships:

MATCH (n)-[r]->() RETURN COUNT(r)

Outgoing relationship with condition:

MATCH (n)-[r]->() WHERE n.flag= 'active' RETURN COUNT(r)

We are using “flag” as one of the properties in legal node to identify whether the law is active or suspended. So that all the users can access the detailed version and evaluation of particular law.

Flag status : query to find flag status

match (p:thread {tag:'law'})-[l:under]->(j:legal_sub_modified {flag:'active'})

return p, l,

We are using “tag” as one of the properties in order to give suggestions to user based. Tags are maintained in tag collections where each and every article is given with a tag Sections recently used for one of the option for suggestion.

match (p:user {profession:'Doctor'})-[rel:dicussion]->(j:thread{thread_id:'55gt4651t65r6gf654gfv61'})

SET j :CacheQuery

return j.tag

As tag is stored in cache we can use it for suggestion if the number of hits for this particular section is high in redis for other users who’s profession is Doctor. In same scenario we can use Age, place tags for suggestions. Our Data model is designed for optimizing fast management, storage, and traversal of nodes and relationships.

Redis

Legal section scoreboard:

Each legal section's Unique mongo ID with a prefix 'l' is store as the key for the sorted dataset (legal section score-board) and the values is a combination of thread unique mongo ID and scores for that thread.

An example of legal section score-board looks like this:

ZADD l<mongoID for legal section> score <mongo ID for thread1 under that legal section> score <mongo ID for thread2 under that legal section>

In-order to retrieve the scoreboard for the scoreboard we used the 'ZREVRANGE' command to get the scores in descending order

An example looks like this:

ZREVRANGE <mongoID for legal section> <range> WITHSCORES

This will return me a set (specified in range) of thred id's and corresponding score from the legalsec-tion scoreboard.

ZINCRBY <legal section mongo ID> <increment(number)> <thread mongo ID> is used to increment a threads rank.

User score board:

The user score board is stored, updated and retrieved the same way as legal section score board. But here we will be having one scoreboard for all user with key as 'userscoreboard' and value pairs as unique mongo ID of the user and their score.

An example looks like this:

ZADD userscoreboard <score> <User1MongoID> <score> <User2MongoID>

The commands are same as Legal section scoreboard..

6.2 User Stories and Queries

User story 1) MongoDB and redis

a) Fetch the legal working hours details from entire laws data with key search with title or word search and relative discussions done upon the thread.

Note: Refer creation of all collections in database from **Appendix B**

db.legalLaws.aggregate([{ $match: { legalSectionTitle: { $regex: /Working Time/} } }, { $lookup: {from: 'threads', localField: '_id', foreignField: 'legalID', as: '<<string>>' }}]).pretty()

(User stories results refer to Appendix - results.doc – fig.5.1)

In order to sort the comments the threads

zrevrange l5d019ba889509535e09592d2 0 100 WITHSCORES

b) Fetch the thread of end-user and relevant discussion performed by users under legal working hours.

db.threads.aggregate([{ $match: { title: { $regex: /addi/ } } }, { $lookup: { from: 'legalLaws', localField: 'legalID', foreignField: '_id', as: '<<string>>' } }]).pretty()

User story 2 MongoDB and redis

a) Admin creates a new document in legalLaws collection as an update for the older law. The older version object ID has to be mentioned in predecessor field and status to be as active of the new docu-ment.

db.legalLaws.insert({"legalSectionIdm":"ArbZG1","legalSectionTitle":"Law regulating a general min-imum wage","legalSubSectionID":"7","legalSubSectionIdm":"7","legalSubSectionTitle":["fixing the general minimum wage","minimum wage"],"description":"(1) Every employee shall be entitled to a salary equal to at least the minimum wage paid by the employ-er..","referanceLink":"http://www.gesetze-im-internet.de/milog/BJNR134810014.html#BJNR134810014BJNG000200000","refering":[""],"createdDate":"2019-06-13T08:29:14.851Z","lawCreatedDate":"1994-06-19","createdby":"5d015b76a9294ed7d423c0e6","predecessor":["5d01b17289509535e09592d6"],"successor":[""],"reason":"minimum wages acts","country":"Germany","imageLink":"","videoLink":"","Active":"Yes"})

b) The older document will be updated with inactive status and new laws object ID in successor and other relevant fields.

db.legalLaws.update({_id:ObjectId("5d01b17289509535e09592d6")},{$push:{successor:{new_refe_obj: "5d03260889509535e09592ef"}},Active:"No"})

c) The scoreboard key name of the inactive one renamed to new object ID.

RENAME l44g01b172807j0535e19d9b7d l44g01b172807j0535e19d9b8d

(User stories results refer to Appendix - results.doc – fig.5.2)

User story 3) MongoDB

a) User creates a thread in the portal.

db.threads.insert({ title:“Could any one please suggest me regarding temporary employment laws in Germany", description: “As I don’t have relevant or present updated information so this could be better for my research project", createdDate: "2019-06-12", legalId: "5d032433def9210c8c6081ea", im-ageLink: “www.abc.com", videoLink:“www..com"})

b) The thread Mongo ID will be added to the scoreboard with initial score.

ZADD l44g01b172807j0535e19d9b7d 0 "5d02781289509535e09592ec"

c) Other users commenting under that discussion.

db.threads.update({_id:ObjectId("5d02525389509535e09592eb")},{$push:{comments:{_id: ObjectId(), description:„According to government official website there is no additional data provided", createdBy:"5d024ef89509535e095920e8" }}})

d) The thread score updated in the legal section scoreboard.

ZADD l44g01b172807j0535e19d9b7d 5 "5d02781289509535e09592ec"

User story 4) Neo4j

User story 4) a) Neo4j

MATCH (:user { user_id:'2g01b172805008jhe959b799' })-[r: follow]->(l: user)

SET l: CacheQuery

RETURN type(r), l.user_id

b)

Match (l:user: CacheQuery)-[rel:dicussion]->(a:thread)

SET a: CacheQuery

RETURN a.thread_id

c)

MATCH (a:thread: CacheQuery)-[:under]->(m:legal)

SET m: CacheQuery

RETURN m.section_id, m.section

d)

MATCH (m:legal: CacheQuery)

MATCH (s: user { user_id:'2g01b172805008jhe959b799' })

RETURN m.section as suggestion_for, s.user_id as user_6

User Story 5

a) Fetch the User scoreboard from Redis

ZREVRANGE userscoreboard 0 5 WITHSCORES

b) Fetch the details of each user from the user collection which is stored in Mongo DB.

```
db.users.find({
  _id: {
    $in: [ObjectId("55880c251df42d0466919268"), ObjectId("55bf528e69b70ae79be35006")]
  }
});
```

User story 6

a) Fetch the corresponding scoreboard for the legal section

ZREVRANGE l5d019ba889509535e09592d2 0 100 WITHSCORES

b) Fetch the details of each threads and comments from the threads collection which is stored in Mongo DB.

```
db.threads.find({
  _id: {
    $in: [ObjectId("5d01b17289509535e09592d6"), ObjectId("5d02525389509535e09592eb")]
  }
});
```

(User stories results refer to Appendix)

7.Conclusion

The discussion form about labor and employment laws data model has been created successfully. This covers the CRUD operations and provides users to have their flexibility of finding their scenarios and search for relevant data about legal laws. Redis is ready with score board creation for the purpose of maintaining the number of hit through MongoBD and Neo4j in order to count.

To conclude the user stories realized the project is fulfilled accordingly. This covers the CRUD operations and provides users to have their flexibility of finding their scenarios and search for relevant data about legal laws. Redis is ready with score board creation for the purpose of maintaining the number of hit through MongoDB and Neo4j in order to count. All the use cases which required whole information of legal laws are effectively utilized as planned. First, second and third use case are fulfilled using MongoDB and Redis. Forth user story connects all three data bases (MongoDB, Neo4j & Redis). The last two use cases are depended mainly on Redis and MongoDB. A major problem in MongoDB was representation of heterogenous files, which are currently shown as source-links. cases were solved using Neo4j, a graphical label, property database. All the relations were successfully mapped in Neo4j with labels. Initially, mapping of entities to nodes was minimal but as the project proceeded, more entities were created as nodes and more relations and properties were added. At last, all the databases were made keeping in mind their purpose and were accordingly used.

8.List of Reference

https://redis.io/documentation

https://docs.mongodb.com/

https://neo4j.com/docs/

https://neo4j.com/docs/cypher-manual/current/clauses/match/

https://businessculture.org/western-europe/business-culture-in-germany/work-life-balance-in-germany/

https://www.howtogermany.com/pages/temp-workers.html

https://www.howtogermany.com/pages/legal.html

9.Appendix A

This contains all script files of mongoDB, Neo4j, redis.

The result documents