

MAP INNOVATIVE
TOPIC NAME: Blog Posting website

PREPARED BY:-MANAN PABARI(17BIT044)
MANISH KHATRI(17BIT045)
JENIL MEHTA(17BIT048)
MEHUL BANSAL(17BIT049)
MEHUL MEHTA(17BIT050)

Introduction

A blog (a shortened version of “weblog”) is an online journal or informational website displaying information in reverse chronological order, with the latest posts appearing first, at the top. Here we have tried to create a blog system, using REST. We have used MongoDB for our database. We have also used Message Queueing in our system. The resulting system allows the users to create, read and delete the blogs efficiently. The system also allows users to edit their blogs. We have also included a navigation bar and user dashboard to navigate through the system.

SERVICES OFFERED

REST SERVICES

WHAT ARE REST SERVICES?

The acronym for REpresentational State Transition is REST. It is an architectural style for distributed hypermedia systems and Roy Fielding first introduced it in his famous dissertation in 2000.

REST also has its own 6 guiding restrictions, like every other architectural design, which must be met if an interface is to be called RESTful. These values are listed below.

Guiding Principles of REST

1. **Client-server**:-By separating user interface problems from data storage issues, we enhance user interface portability across multiple platforms and enhance scalability by simplifying the components of the server.
2. **Stateless**: Each client-to-server request must include all the information required to understand the request and can not take

advantage of any context stored on the server. Therefore the session state is held exclusively by the client.

3. **Cacheable**-Cache constraints require that the data be implicitly or specifically classified as cacheable or non-cacheable within a response to a request. If a response is cacheable, then the right to reuse the response data for subsequent, similar requests is provided to a client cache.
4. **Uniform interface**- The overall device architecture is streamlined and the visibility of interactions is enhanced by applying the software engineering concept of generality to the component interface. Multiple architectural limitations are needed to direct the behaviour of components in order to achieve a uniform interface. Four interface constraints describe REST: resource identification; resource exploitation by representations; self-descriptive messages; and, hypermedia as the engine of application state.
5. **Layered system**-The layered system style makes it possible for an architecture to consist of hierarchical layers by restricting component behaviour so that each component does not see" beyond the immediate layer in which they communicate.
6. **Code on demand** (optional)-REST enables client functionality to be expanded in the form of applets or scripts by downloading and executing code. This simplifies consumers by reducing the amount of pre-implemented features available.

Resource

A resource is the main abstraction of knowledge in REST. A resource can be any information that can be named: a document or photo, a temporary service, a set of other resources, a non-virtual entity (e.g. a person), etc. To define the specific resource involved in an interaction between components, REST uses a resource identifier.

The condition of the resource is known as resource representation at any given timestamp. A representation consists of content, data describing metadata and hypermedia connections that can assist customers in the transition to the next desired state.

As a media type, the data format of a representation is known. A specification describing how a representation is to be processed is defined by the media type. A genuinely RESTful API is hypertext-like. An address is held by any addressable unit of knowledge, either directly (e.g. link and id attributes) or indirectly (e.g. link and id attributes) (e.g., derived from the media type definition and representation structure).

In addition, resource representations are self-descriptive: the client does not need to know whether the employee or computer is a resource. It should function on the basis of the form of media associated with the resource. So you will end up creating loads of custom media types in practise usually one media type linked to one resource.

Resource Methods

Resource methods to be used to perform the desired transformation are another essential thing connected with REST. A significant number of people incorrectly compare resource methods to the methods of HTTP GET/PUT/POST/DELETE.

Ideally, anything needed to alter the resource state would be part of that resource's API response, including methods and in what state they will leave the representation. .

Another thing that will help you create RESTful APIs is that the results of the query-based API should be represented by a list of links with summary information, not by arrays of original resource representations, as the query is not a replacement for resource recognition.

REST and HTTP are not same !!

A lot of people prefer to compare HTTP with REST. **REST and HTTP are not same.**

REST != HTTP

While REST also aims to streamline and standardise the web (internet), he advocates more strictly using REST concepts. And that's where people are trying to start comparing REST with the Internet (HTTP). In his dissertation, Roy Fielding did not discuss any implementation directive, including any preference for protocols and HTTP. You should call your interface RESTful until the moment you respect the 6 guiding principles of REST.

In simple terms, data and functionality are called resources in the REST architectural style, and are accessed using Uniform Resource Identifiers (URIs). By using a set of simple, well-defined operations, the resources are acted upon. By using a structured interface and protocol, usually HTTP, the clients and servers share representations of resources.

In order to access their content in a number of formats, such as HTML, XML, plain text, PDF, JPEG, JSON, and others the tools are decoupled from their representation. Resource metadata is available and used for example, to control caching, detect errors in transmission, negotiate the appropriate format of representation, and perform authentication or access control. And most significantly, any interaction is stateless with a resource.

All these principles help RESTful applications to be simple, lightweight, and fast.

DATABASE:- MONGODB

What is MongoDB?

MongoDB is a NoSQL document-oriented database used for data storage of high volumes. MongoDB makes use of sets and records instead of using

tables and rows, as in conventional relational databases. Documents consist of key-value pairs in MongoDB that are the basic data unit. The collections contain documents and feature sets that are similar to relational database tables. A database that came into light in the mid-2000s is MongoDB.

MongoDB Features

1. There are collections in each database, which in turn contain records. With a varying number of fields, each document may be different. Each document's size and content can be different from each other.
 2. The document structure is more in line with how developers in their respective programming languages build their classes and objects. Developers would also say that their groups are not rows and columns, but that key-value pairs have a simple structure.
 3. The rows (or documents referred to in MongoDB) do not need to have a schema specified in advance. Instead it is possible to build fields on the fly.
 4. Within MongoDB, the data model available enables you to more easily represent hierarchical relationships, store arrays, and other more complex structures.
1. Scalability – The MongoDB environments are very scalable. Companies across the world have defined clusters with some of them running 100+ nodes with around millions of documents within the database

MongoDB Example

The below example shows how a document can be modeled in MongoDB.



1. The `_id` field is added by MongoDB to uniquely identify the document in the collection.
2. What you can note is that the Order Data (OrderID, Product, and Quantity) which in RDBMS will normally be stored in a separate table, while in MongoDB it is actually stored as an embedded document in the collection itself. This is one of the key differences in how data is modeled in MongoDB.

```
{
  _id : <ObjectId> ,
  CustomerName : Guru99 ,
  Order:
    {
    }
}
```

OrderID: 111
Product: ProductA
Quantity: 5

Example of
how data can
be embedded
in a document

Key Components of MongoDB Architecture

Below are a few of the common terms used in MongoDB

1. **`_id`** – This is a field required in every MongoDB document. The `_id` field represents a unique value in the MongoDB document. The `_id` field is like the document's primary key. If you create a new document without an `_id` field, MongoDB will automatically create the field. So for example, if we see the example of the above customer table, Mongo DB will add a 24 digit unique .
2. **Collection** – This is a grouping of MongoDB documents. A collection is the equivalent of a table which is created in any other RDMS such as Oracle or MS SQL. A collection exists within a single database. As seen from the introduction collections don't enforce any sort of structure.

3. **Cursor** – This is a pointer to the result set of a query. Clients can iterate through a cursor to retrieve results.
4. **Database** – This is a container for collections like in RDMS wherein it is a container for tables. Each database gets its own set of files on the file system. A MongoDB server can store multiple databases.
5. **Document** - A record in a MongoDB collection is basically called a document. The document, in turn, will consist of field name and values.
6. **Field** - A name-value pair in a document. A document has zero or more fields. Fields are analogous to columns in relational databases.
The following diagram shows an example of Fields with Key value pairs. So in the example below CustomerID and 11 is one of the key value pair's defined in the document.

Why Use MongoDB?

Below are the few of the reasons as to why one should start using MongoDB

1. Document-oriented – Since MongoDB is a NoSQL type database, instead of having data in a relational type format, it stores the data in documents. This makes MongoDB very flexible and adaptable to real business world situations and requirements.
2. Ad hoc queries - MongoDB supports searching by field, range queries, and regular expression searches. Queries can be made to return specific fields within documents.
3. Indexing - Indexes can be created to improve the performance of searches within MongoDB. Any field in a MongoDB document can be indexed.
4. Replication - MongoDB can provide high availability with replica sets. A replica set consists of two or more mongo DB instances. Each replica set member may act in the role of the primary or secondary replica at any time. The primary replica is the main server which interacts with the client and performs all the read/write operations. The Secondary replicas maintain a copy of the data of the primary

using built-in replication. When a primary replica fails, the replica set automatically switches over to the secondary and then it becomes the primary server.

5. Load balancing - MongoDB uses the concept of sharding to scale horizontally by splitting data across multiple MongoDB instances. MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure.

Data Modelling in MongoDB

As we have seen from the Introduction section, the data in MongoDB has a flexible schema. Unlike in SQL databases, where you must have a table's schema declared before inserting data, MongoDB's collections do not enforce document structure. This sort of flexibility is what makes MongoDB so powerful.

When modeling data in Mongo, keep the following things in mind

1. What are the needs of the application – Look at the business needs of the application and see what data and the type of data needed for the application. Based on this, ensure that the structure of the document is decided accordingly.
2. What are data retrieval patterns – If you foresee a heavy query usage then consider the use of indexes in your data model to improve the efficiency of queries.
3. Are frequent inserts, updates and removals happening in the database? Reconsider the use of indexes or incorporate sharding if required in your data modeling design to improve the efficiency of your overall MongoDB environment.

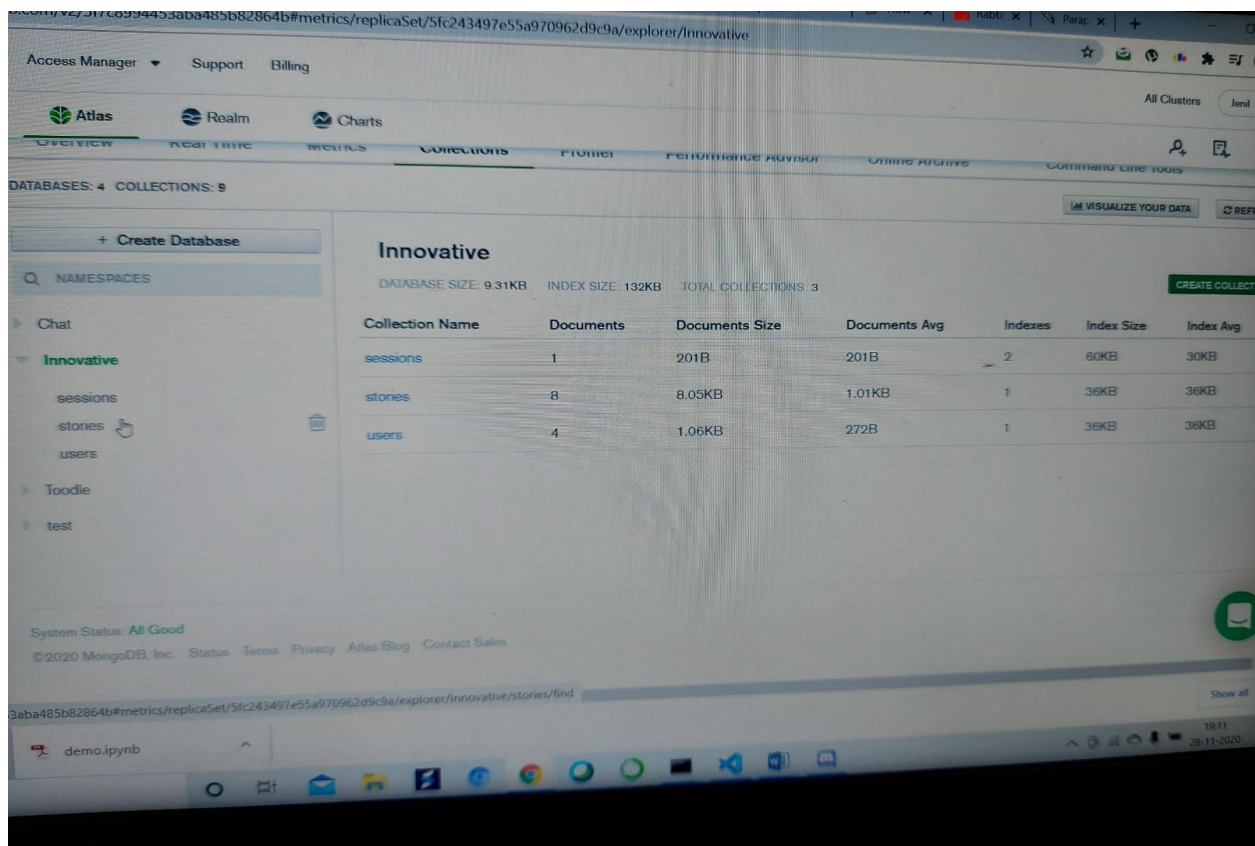
Difference between MongoDB & RDBMS

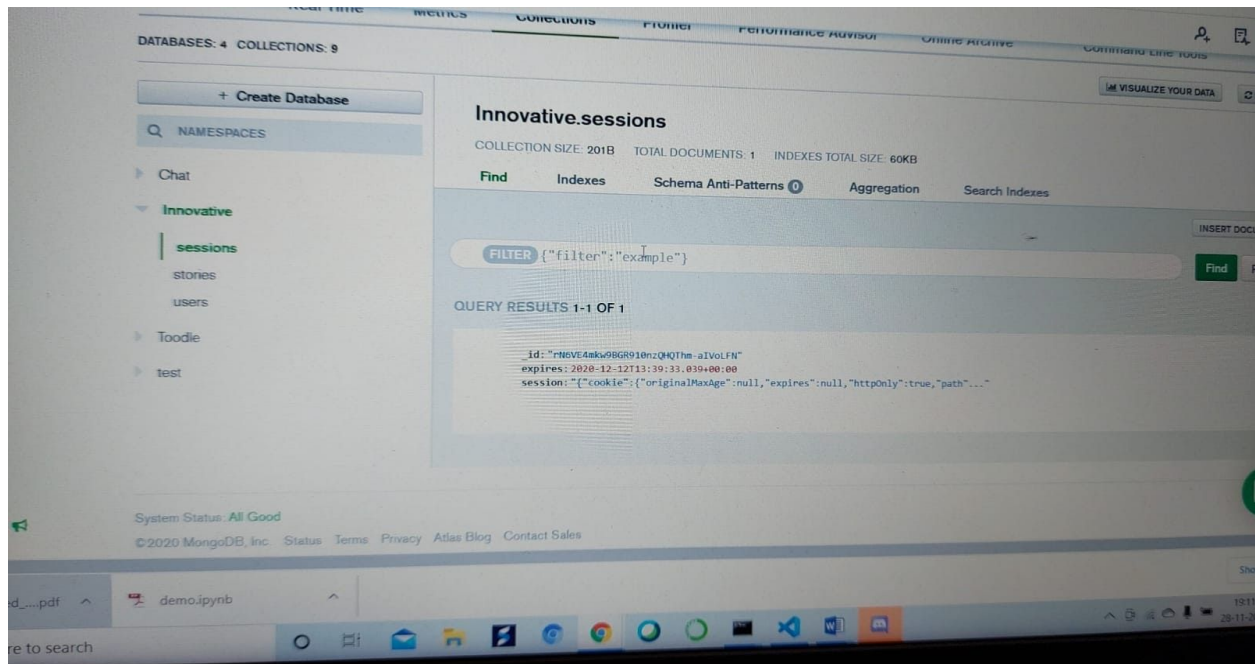
Below are some of the key term differences between MongoDB and RDBMS.

RDBMS	MongoDB	Difference
Table	Collection	In RDBMS, the table contains the columns and rows which are used to store the data whereas, in MongoDB, this same structure is known as a collection. The collection contains documents which in turn contains Fields, which in turn are key-value pairs.
Row	Document	In RDBMS, the row represents a single, implicitly structured data item in a table. In MongoDB, the data is stored in documents.
Column	Field	In RDBMS, the column denotes a set of data values. These in MongoDB are known as Fields.
Joins	Embedded documents	In RDBMS, data is sometimes spread across various tables and in order to show a complete view of all data, a join is sometimes formed across tables to get the data. In MongoDB, the data is normally stored in a single collection, but separated by using Embedded documents. So there is no concept of joins in MongoDB.

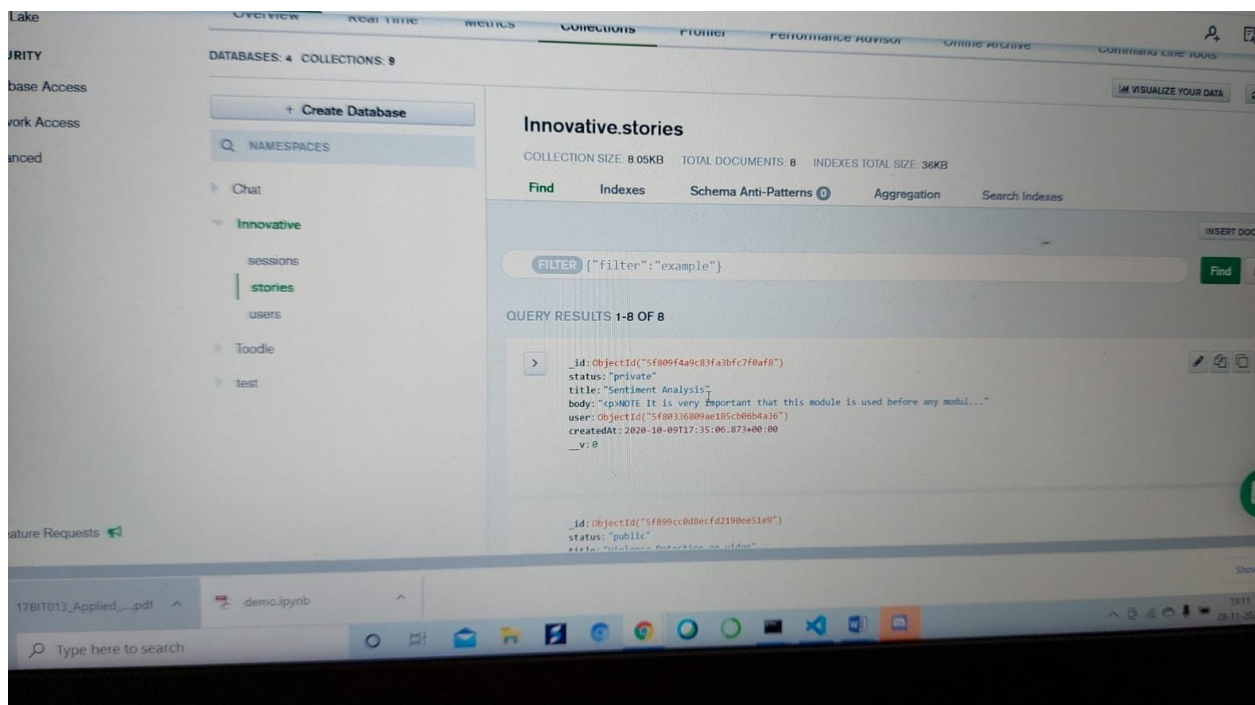
Our Schema:

1. Sessions
2. Stories
3. Users.

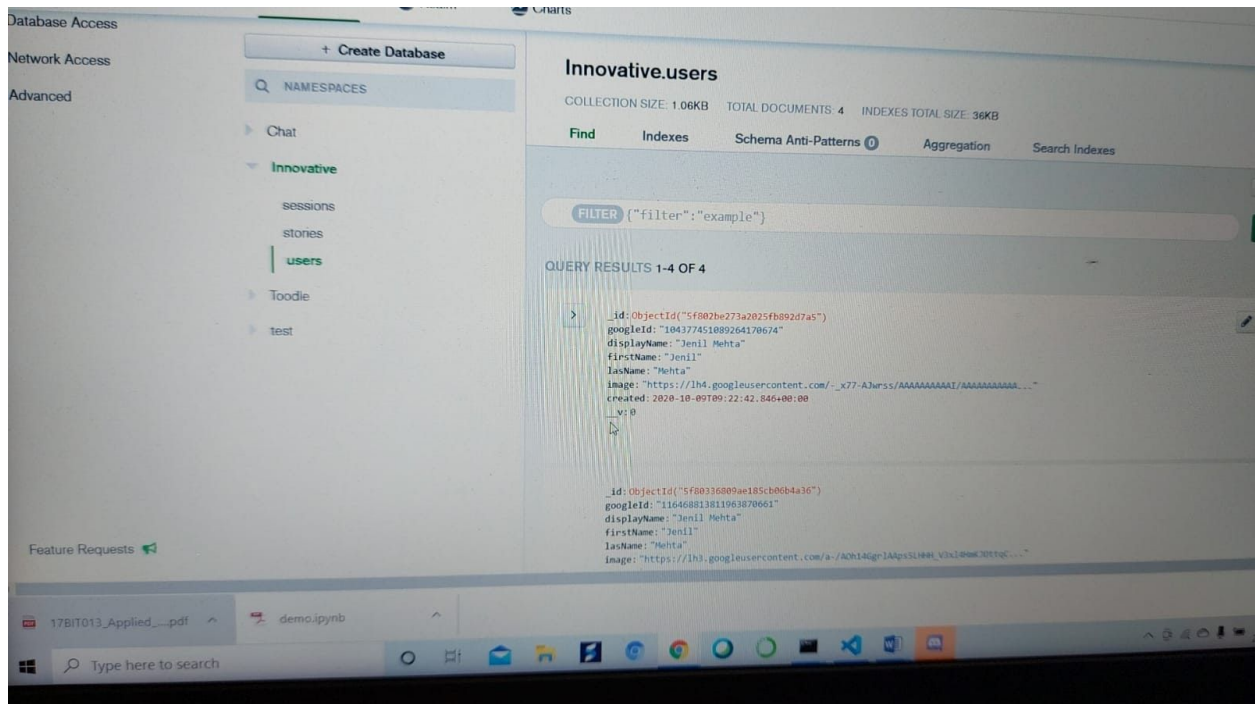




This contains details about current sessions of logged in users. The attributes present in the collections are ID, Expires, Session.



The stories collection consist of details about the status, title, userId, content and date of creation.



User collection will consist of details about users that have registered to our system.

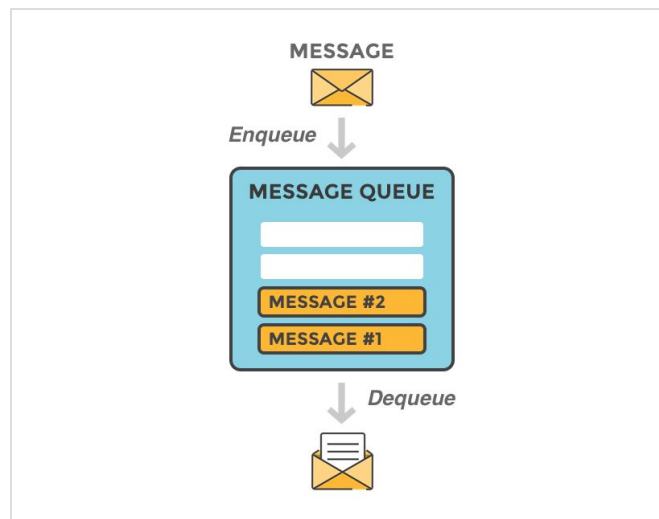
MessageQueue

A message queue is a form of asynchronous service-to-service communication used in serverless architectures and microservices. In the queue, messages are stored until they are processed and removed. Each message is handled by a single user, and only once. To decouple heavyweight processing, buffer or batch work, and smooth spiky workloads, message queues can be used.

Message queuing helps users to connect by sending messages to each other. The message queue includes a temporary queue for messages

Example of Message Queue

A queue is a line of items waiting to be dealt, starting in sequential order at the beginning of the line and processing it. A message queue is a queue of messages that are sent between apps. This requires a collection of functioning items that are waiting to be processed. A message is the data carried between the sender and the application of the receiver; it is simply a byte array with some headers at the top. An example of a message may be something that tells a machine to begin processing a task, may contain information about a completed task, or may just be a simple message.



A message queue's basic architecture is simple; there are client applications called producers that generate messages and distribute them to the queue of messages. Another program called a customer, links to the queue and gets the messages to be processed. Messages entered in the queue are stored until they are retrieved by the user.

Benefits of message queue

Applications are decoupled into smaller, autonomous building blocks in modern cloud architecture that are easier to design, deploy and manage. For these distributed applications, message queues provide connectivity and coordination.

Message queues, while improving efficiency, reliability and scalability, will greatly simplify the coding of decoupled applications. You may also merge message queues in a fan-out style pattern with Pub/Sub messaging.

RabbitMQ

A message broker acts as a middleman for various services (e.g. a web application, as in this example). They can be used to reduce loads and delivery times of web application servers by delegating tasks that would normally take up a lot of time or resources to a third party that has no other job.

We pursue a situation in this guide where a web application makes it possible for users to upload information to a website. The site will handle this data, create a PDF and send it back to the user by email. In this example, handling the details, generating the PDF, and sending the email will take several seconds. That is one of the reasons why the task will be performed using a message queue.

The web application will generate a "PDF processing" message when the user has entered user information into the web interface, which will contain all the important information that the user needs in a message and position it on a queue specified in RabbitMQ.



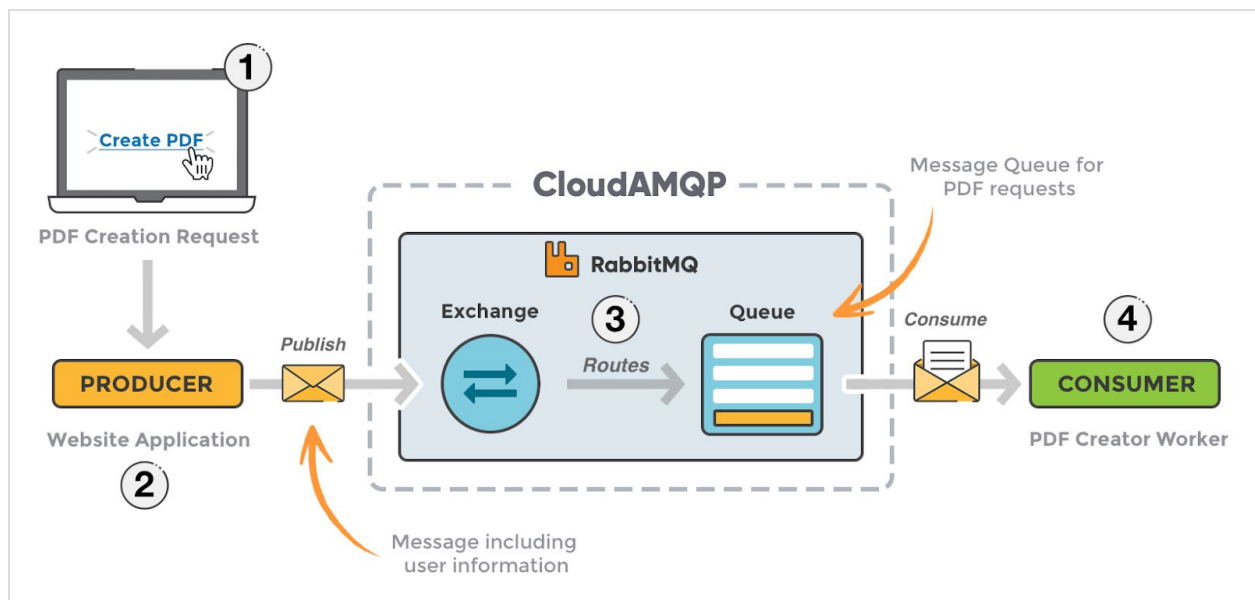
The basic message queue design is simple - there are client apps called producers that generate messages and deliver them to the broker (the message queue). Other users, referred to as customers, connect to the queue and subscribe to the messages to be processed. Software can function as a producer of messages, or as a consumer, or as both a consumer and a message producer. Messages entered in the queue are stored until they are retrieved by the user.

When and why should you use RabbitMQ?

Instead of being required to conduct resource-heavy procedures on the spot that can prolong response time, message queueing enables web servers to respond to requests quickly. When you want to distribute a

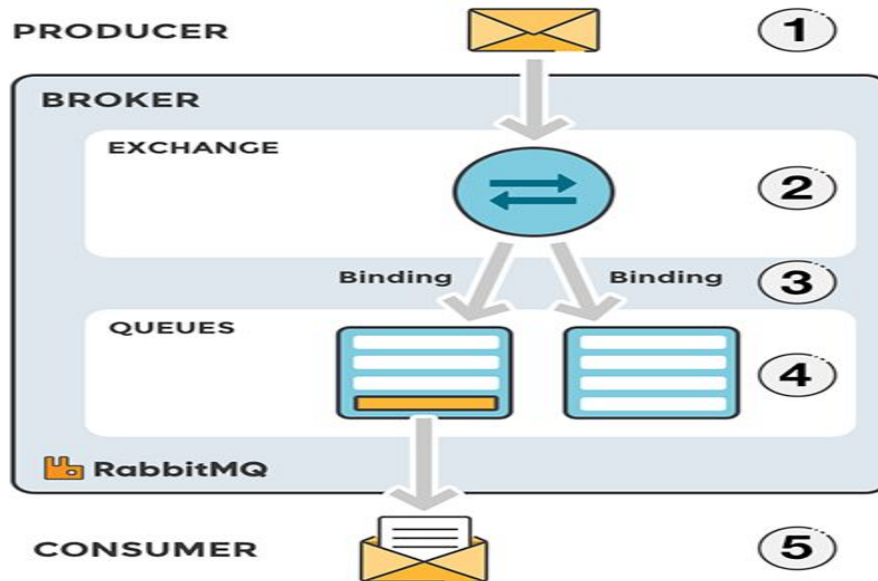
message to multiple customers or to balance loads between employees, message queueing is also good.

The customer takes a message from the queue and begins to process the PDF. At the same moment, new messages are being queued up by the manufacturer. The user can be on a server that is entirely different from the manufacturer, or they can be placed on the same server. You can create the request in one programming language and handle it in a different programming language. The argument is the two applications can communicate only through the messages they send to each other, which means that there is low coupling of the sender and recipient.



1. The user sends a PDF creation request to the web application.
2. The web application (the producer) sends a message to RabbitMQ that includes data from the request such as name and email.
3. An exchange accepts the messages from the producer and routes them to correct message queues for PDF creation.
4. The PDF processing worker (the consumer) receives the task message and starts processing the PDF.

Message Flow in RabbitMQ

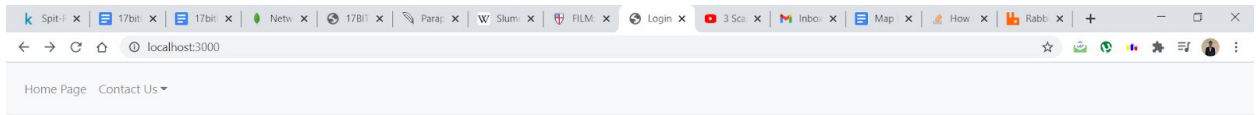


1. The producer publishes a message to an exchange. When creating an exchange, the type must be specified. This topic will be covered later on.
2. The exchange receives the message and is now responsible for routing the message. The exchange takes different message attributes into account, such as the routing key, depending on the exchange type.
3. Bindings must be created from the exchange to queues. In this case, there are two bindings to two different queues from the exchange. The exchange routes the message into the queues depending on message attributes.
4. The messages stay in the queue until they are handled by a consumer
5. The consumer handles the message.

Working of our project:

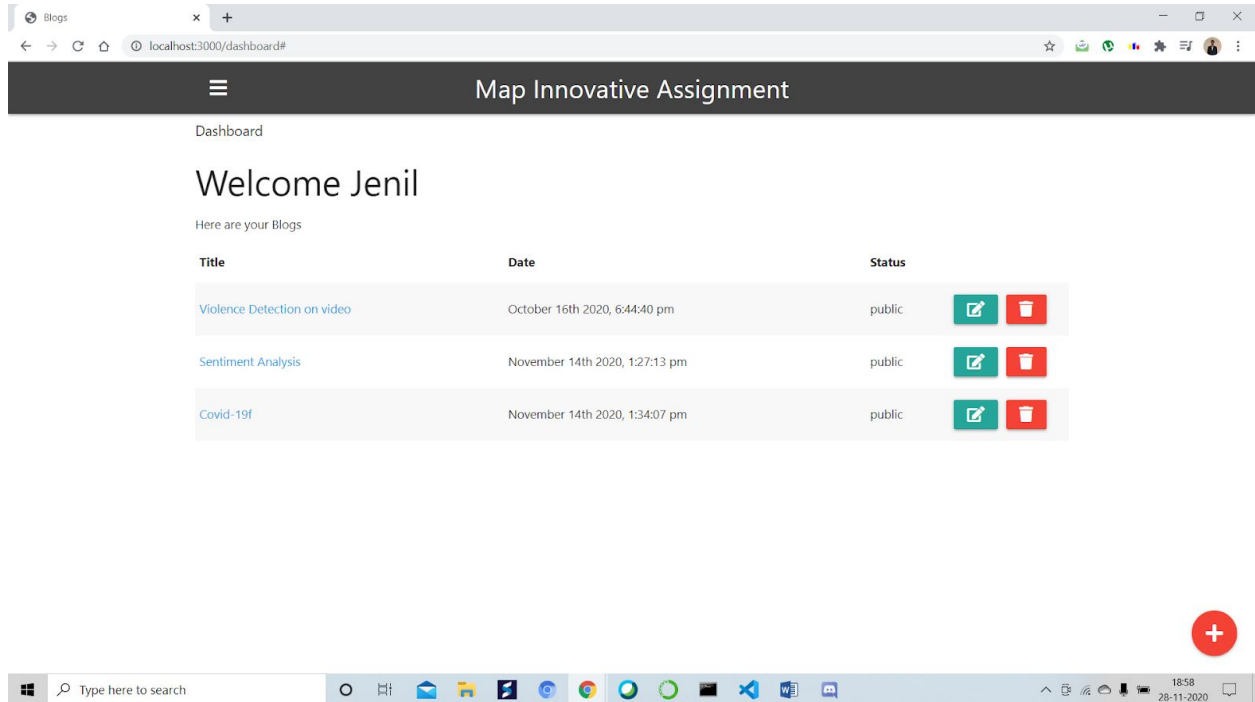
1. Login:

The user needs to be logged in the website using google login. The user if not registered will be registered and the username will be added into the database.



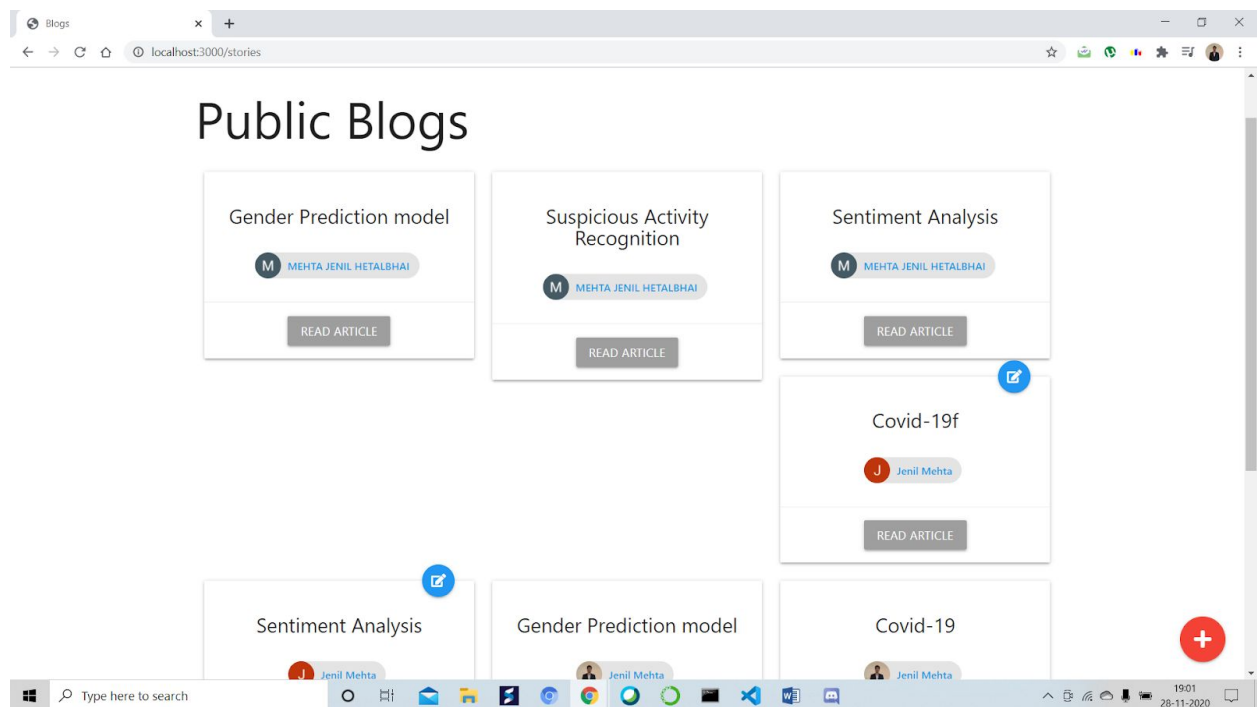
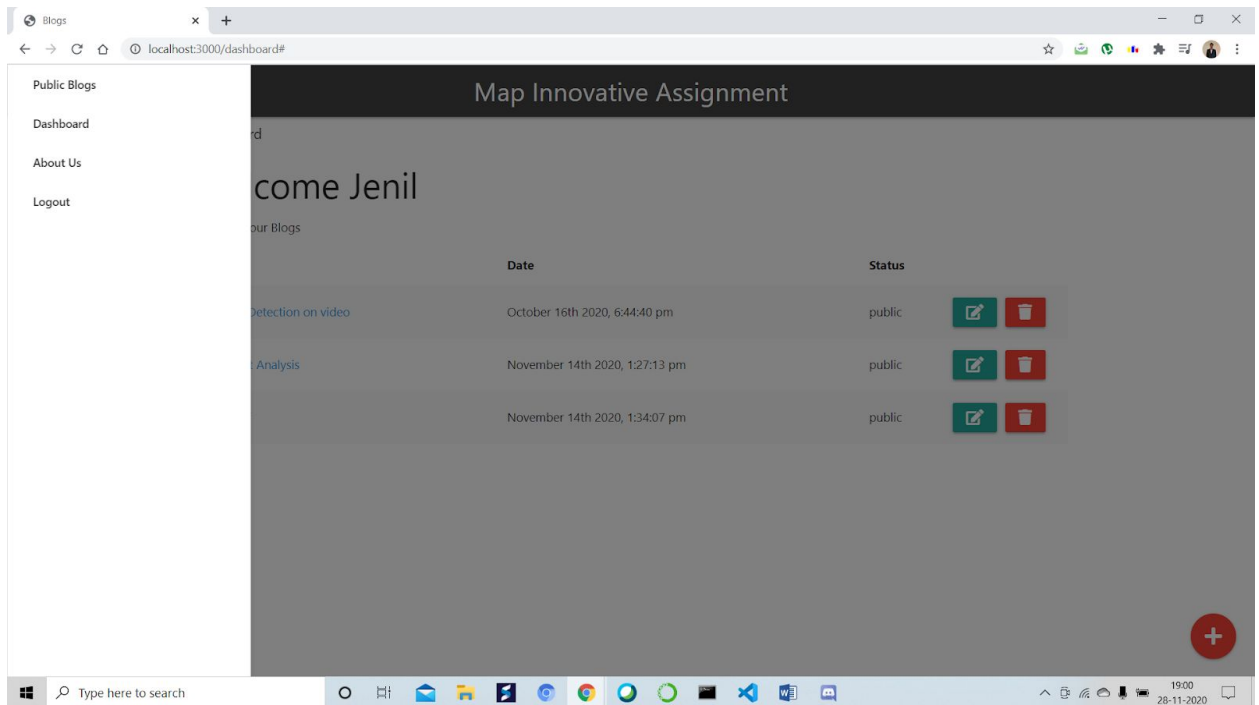
2. Dashboard:

Once logged in the user will be directed to the dashboard page. This page will show all public blogs that are written by the current user. The info displayed are Title of the blog, Date of being posted, Status(public/private).



3. Side Nav-bar:

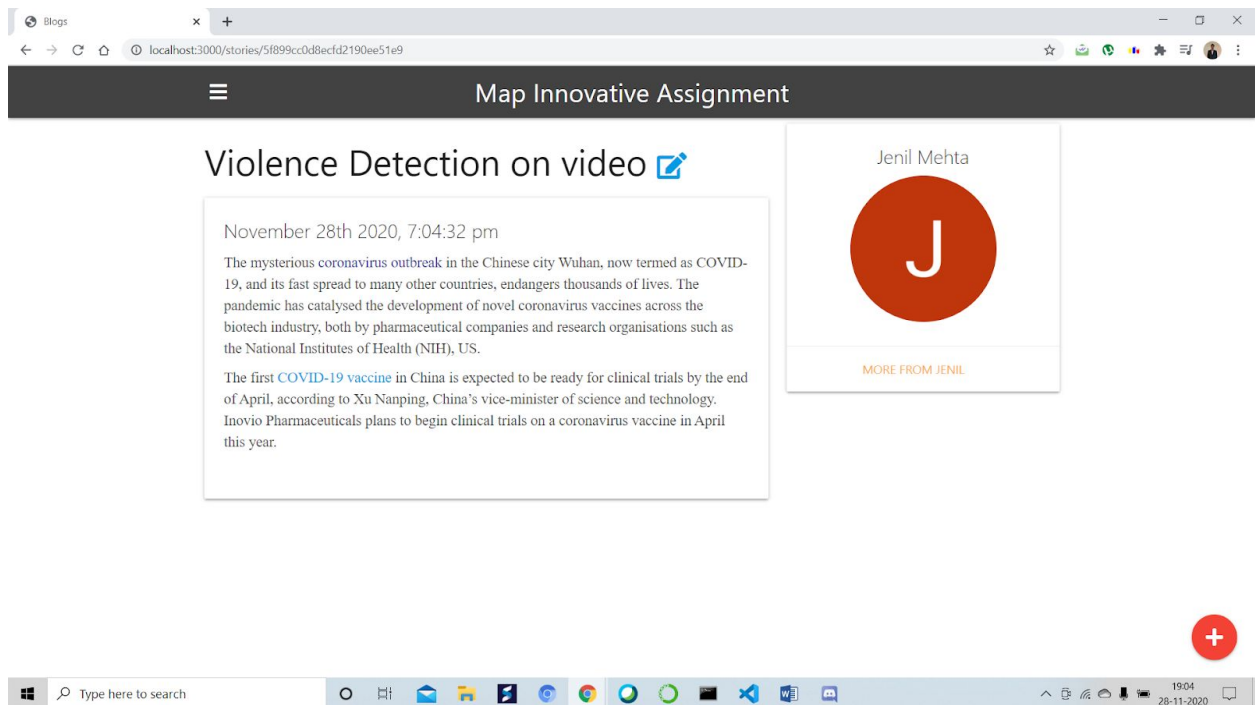
There are Public Blogs, Dashboard, About Us and logout. Dashboard is already explained. Public blogs will contain all the public blogs written by all registered users.



Here as we can see the user can edit the blogs that were written by him/her. He can only view the blogs of other registered users.

4. Read article:

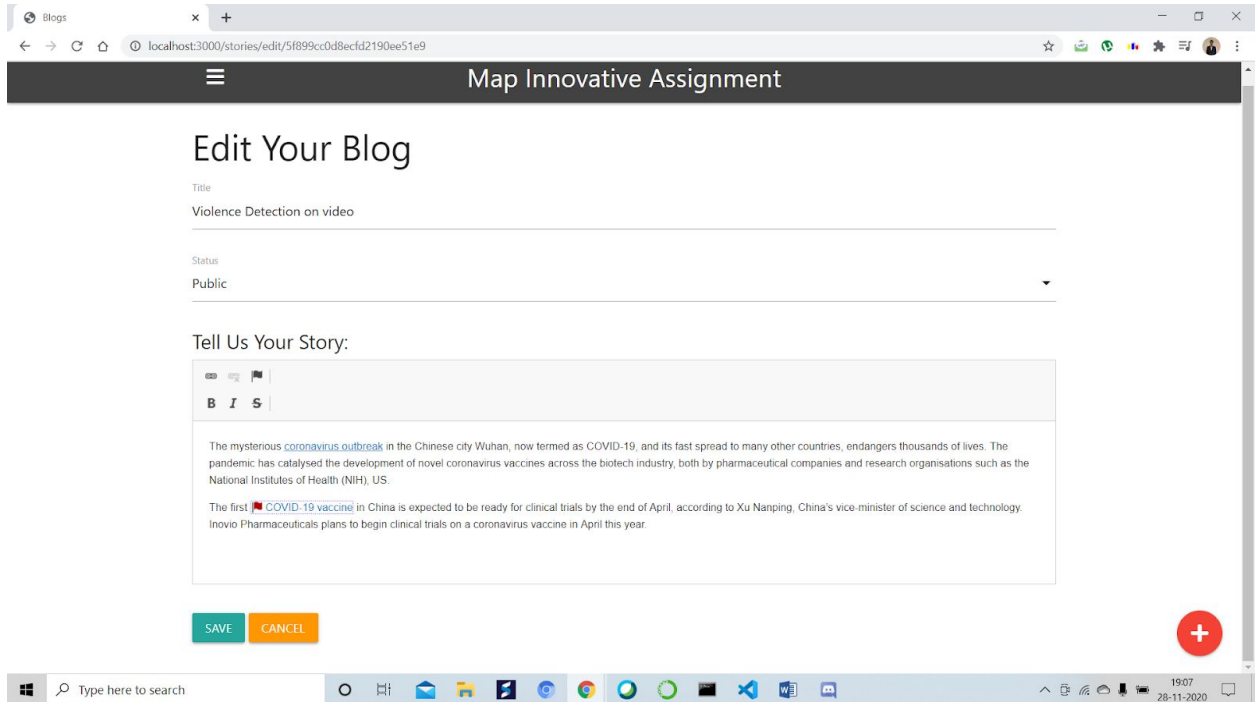
Once a user clicks on a read article on any user's story he will be shown the entire article.



Here there are several other options that users can explore.

5. Edit and Delete:

In the dashboard when a user clicks on the edit button he will be diverted to the edit page where he can edit the blog. On the edit page the user can change entire blog.







Here we have a good editor to highlight some text like italic, bold, link some text and other options.

6. Delete:

When the user clicks on the delete icon the given story will be deleted from the database. The blog will be edited in the database.

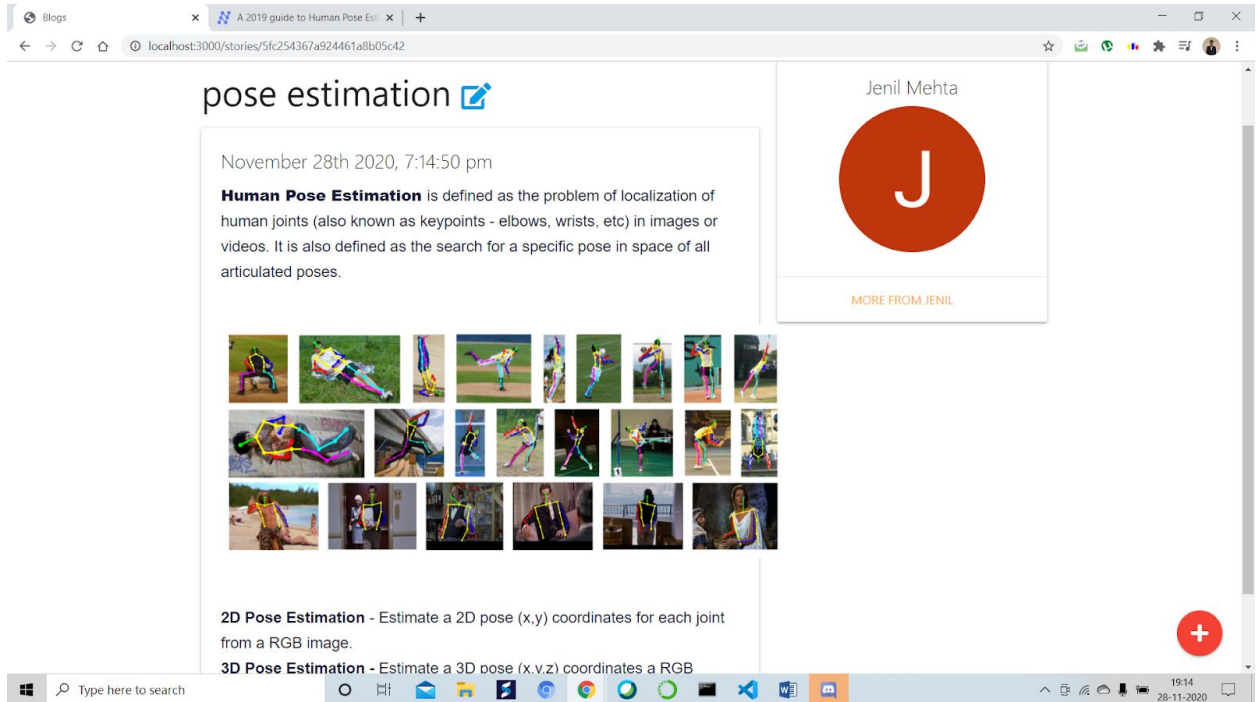
The screenshot shows a web browser window with the address bar displaying 'localhost:3000/dashboard'. The page has a dark header with a hamburger menu icon and the title 'Map Innovative Assignment'. Below the header, the word 'Dashboard' is displayed. A large heading 'Welcome Jenil' is followed by the text 'Here are your Blogs'. A table lists two blogs:

Title	Date	Status	
Violence Detection on video	October 16th 2020, 6:44:40 pm	public	 
Sentiment Analysis	November 14th 2020, 1:27:13 pm	public	 

A red circular button with a white plus sign is located in the bottom right corner of the dashboard area. The Windows taskbar at the bottom shows the search bar and various application icons.

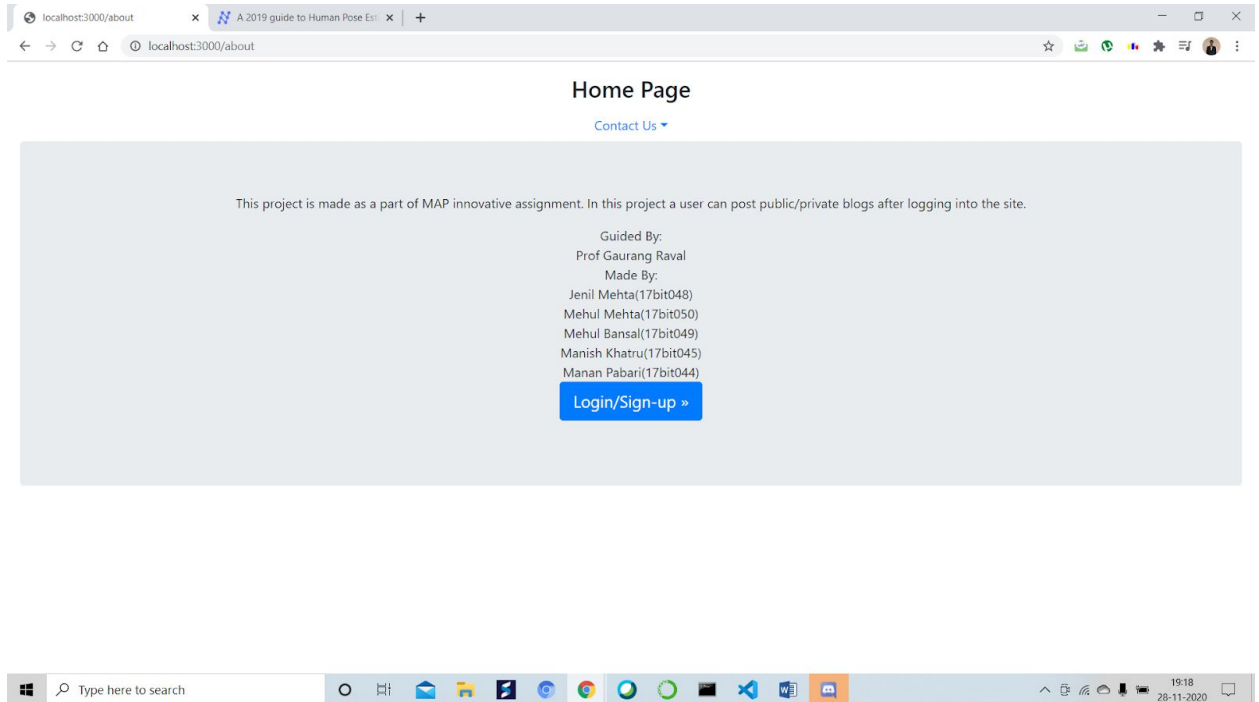
7. Add blogs:

The screenshot shows the 'Add Story' form in the 'Map Innovative Assignment' application. The browser address bar shows 'localhost:3000/stories/add'. The form includes a 'Title' text input field, a 'Status' dropdown menu currently set to 'Public', and a 'Tell Us Your Story:' section with a rich text editor. The rich text editor has a toolbar with icons for bold, italic, and strikethrough, and a text area below it. At the bottom of the form are 'SAVE' and 'CANCEL' buttons. A red circular button with a white plus sign is in the bottom right corner. The Windows taskbar at the bottom shows the search bar and various application icons.



The user can post new blogs in public or private mode. The blog will then be added to the database.

8. Home Page:



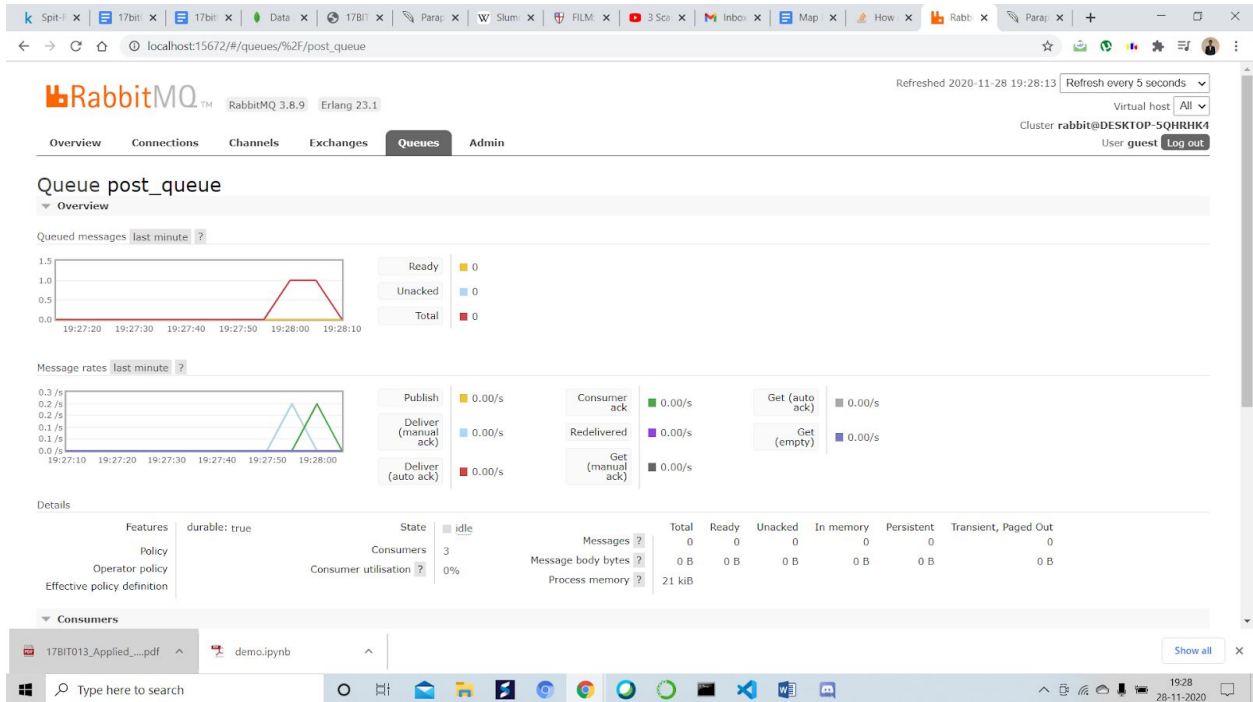
This is just a home page.

9. Sessions:

The user will not be asked to login again and again. The current user session would be stored inside the database. Once the user logs out, he will not be able to access the functionality if he/she is not registered.

10. Message queue:

Once the user posts a new blog the message queue would be implemented.



What is Node.js?

- Node.js is an open source server environment
- Node.js is free

- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js uses JavaScript on the server

Why Node.js?

Node.js uses asynchronous programming!

A common task for a web server can be to open a file on the server and return the content to the client.

Here is how PHP or ASP handles a file request:

1. Sends the task to the computer's file system.
2. Waits while the file system opens and reads the file.
3. Returns the content to the client.
4. Ready to handle the next request.

Here is how Node.js handles a file request:

1. Sends the task to the computer's file system.
2. Ready to handle the next request.
3. When the file system has opened and read the file, the server returns the content to the client.

Node.js eliminates the waiting, and simply continues with the next request.

Node.js runs single-threaded, non-blocking, asynchronous programming, which is very memory efficient.

What Can Node.js Do?

- Node.js can generate dynamic page content

- Node.js can create, open, read, write, delete, and close files on the server
 - Node.js can collect form data
 - Node.js can add, delete, modify data in your database
-

What is a Node.js File?

- Node.js files contain tasks that will be executed on certain events
- A typical event is someone trying to access a port on the server
- Node.js files must be initiated on the server before having any effect
- Node.js files have extension ".js"

Conclusion:

So as explained, we used the mentioned technologies to create an application that allows the users to create, delete and edit their blogs and helps them to efficiently navigate through the system.

