# Technical Specifications

The basic architecture of the site would be based on REST API. This will be helpful in future where we can easily extend our site to mobile apps, mobile sites etc. and also we can share our data to third party people by exposing the REST API to the world.

### Frameworks/Libraries Used

* spring-boot to bootstrap the application
* spring-mvc for creating Restful service
* spring-data-jpa to connect to the database layer
* Junit - For unit and integration testing
* Database - MySQL
* Frontend/UI - HTML5, CSS3, Angular JS
* Jenkins - for continuous deployment
* AWS EC2 instance for running the application
* Github for hosting the code so that others can easily collaborate
* Swagger for REST API documentation
* ??? How to create role based UI???
* ~~Good to have featured multiple language support.~~

### REST Specifications

In REST, we have Nouns and Verbs.

Nous are the resources that make the system e.g. user, ticket etc.

Verbs are basically the operations we can perform on resources e.g. GET, POST etc.

#### Verbs/Operations

At start we will be supporting following operations/verbs -

|  |  |
| --- | --- |
| GET | Gets a resource |
| POST | Creates a resource |
| PUT | Updates a resource |
| DELETE | Deletes a resource |
| *PATCH* | *Partially updates a resource (future addition)* |

#### Nouns/Resources

The system will have following resources -

User

It can have following three roles

* **ADMIN** - This role has access to all the system resources. It can add/delete/approve other users. It can approve/reject other user’s articles. Basically it is the owner of the system and maximum 2 can exist in system.
* **REGISTERED** - Any user who registers to our website will have this role by default. He can publish articles; suggest edits on other’s articles. All articles submitted by this should be approved by ADMIN to be shown on site.

*Future Addition – We can have another role in between ADMIN and REGISTERED. That role will have more permissions than REGISTERED user and any article/links submitted by that can be auto approved. This can be called as VERIFIED user. Later we can put a limit of 10 articles submitted by REGISTERED role if approved by ADMIN, then he can be promoted to VERIFIED user as we know this user is adding valuable info in the system and not spam.*

* **GUEST** - This role can view articles on the site, he can submit articles which can be approved later by ADMIN. He has to give some info e.g. name, email id, mobile number while submitting the article so that later we can approach if required. This role can be used if user want to submit only few articles.

User can have the following fields –

|  |  |
| --- | --- |
| **Field** | **Description** |
| Username | A unique username |
| Password | Password of the user |
| First Name | First name of the user |
| Last Name | Last name of the user |
| Email | Email of the user |
| Phone | Phone number of the user – **may be not mandatory??** |
| Userstatus | Status of the registered user **PENDING, VERIFIED, BLOCKED.** |
| Role | It can be **ADMIN**, **REGISTERED**, **GUEST** |
| Created At | Date when user was created |
| Updated At | Date when user was updated |

##### Article

This is the main content of the site. This is detail about the categories we have already discussed.

Tourism, Administrative, Healthcare, Commercial, Hotel & Transport, Education, Advertisement

An article can have following fields –

|  |  |
| --- | --- |
| **Field** | **Description** |
| Title | Title of article |
| Description | Detail description about the article |
| Category | One or more categories from the above list |
| Tags | One or more tags related to the article, we can use these for search criteria later |
| Image | One or more images about the article |
| Created By | User Id who created the article |
| Approved | Flag to signify if article is approved or not |
| Created At | Date when article was created |
| Updated At | Date when article was updated |

#### Versioning

The system would use URI scheme for versioning the REST API. In this scheme, we embed the version in the URI itself e.g. example.com/v1/users.

This is the easiest way for versioning and client can easily know which version they are using.

#### URIs

**For User**

|  |  |  |
| --- | --- | --- |
| **HTTP Verb** | **URI** | **Detail** |
| GET | /users | Retrieves a list of users |
| GET | /users/2 | Retrieves a specific user #2 |
| POST | /user | Creates a new user |
| PUT | /user/2 | Updates user #2 |
| DELETE | /user/2 | Deletes user #2 |

**For articles created by user**

|  |  |  |
| --- | --- | --- |
| **HTTP Verb** | **URI** | **Detail** |
| GET | /users/2/articles | Retrieves a list of articles by user #2 |
| GET | /users/2/articles/5 | Retrieves a specific article #5 by user #2 |
| POST | /user/2/articles | Creates a new article by user #2 |
| PUT | /user/2/articles/5 | Updates article #5 by user #2 |
| DELETE | /user/2/articles/5 | Deletes article #5 for user #2 |

**For Article**

|  |  |  |
| --- | --- | --- |
| **HTTP Verb** | **URI** | **Detail** |
| GET | /articles | Retrieves a list of articles |
| GET | /articles/4 | Retrieves a specific article #4 |
| POST | These are covered in above table as only a user can add/update/delete an article | |
| PUT |
| DELETE |