Saral Backend Setup:

Prerequisites:

Tech Stack:

- NodeJs

- JavaScript
- AWS
- MongoDB
- Docker
- Jenkins
- Kubernetes
- Github

Suggested Team composition:

Type	Language / Skills	Experience	Full-time people
Backend Developer	NodeJs, JavaScript, Express.js,Postman, MongoDB Compass, SonarQube, Github, Jest Unit testing, Swagger documentation	3+ years	1
DevOps	Docker, AWS, KUBERNETES,jenkins,SCM tools, Grafana, Prometheus,EFK stack, Terraform	3+ years	1
QA	UAT	3+ years	1
Tech Lead	All of the above, & management skills	5+ years	1
Frontend Developer	React Native	3+ years	1
AI	Based on need	3+	1

Dependency Software to install:

- **NodeJS (Recommended version v16.9.1 LTS)**
  - **Windows:** Refer https://nodejs.org/en/blog/release/v16.13.0
  - **Linux:**

```
cd ~

curl -sL https://deb.nodesource.com/setup_16.x -o nodesource_setup.sh

sudo bash nodesource_setup.sh

sudo apt-get install nodejs

node -v
```

  - After install verify nodejs version is 16 by typing “node -v” in CLI
- **Mongodb database(https://www.mongodb.com/try/download/community\ )**
  - After install verify mongodb is properly installed by typing “mongod --version” in CLI
- **Docker**
  - After dinstall verify mongodb is properly installed by typing “docker --version” in CLI
- **Git**
  - After install verify mongodb is properly installed by typing “git --version” in CLI

Other software to install for development

- **VoTT tool:** Microsoft VoTT(Visual Object Tagging Tool) VoTT Project Setup Video
- **Jupyter Notebook:** Jupyter Notebook for transforming VoTT raw json to Saral ROI json. Refer to source repository path **specs\v1\README.md** file for more details. You may consider installing the extension in MS Visual Studio Code.
- **Any IDE (Integrated Development Environment) like** MS Visual Studio Code.
- **Postman: https://www.postman.com/downloads/**
- **Mongodb compass**

Local/development setup:

Hardware requirements:

We recommend following hardware requirements as minimum requirements to setup Saral backend.

One server is required with the following configurations:

- Any OS
- 16 GB of System RAM (minimum requirement)
- 4 core CPU (minimum requirement)
- 250GB HDD

Network Ports requirement:

Local deployment/on premises:

- 3000(required to launch node server)
- 27017(to launch mongodb)

Installation/Setup:

Setting up saral backend for development:

- Open Terminal and clone source code “git clone https://github.com/Sunbird-Saral/Project-Saral.git”
- Check latest release https://github.com/Sunbird-Saral/Project-Saral/releases.
- Change Directory to Project-Saral/ folder and switch to release tag as per release notes. git checkout tags/<tag_name>
- Go to folder path “Project-Saral/v1.0/backend”.
- From Saral release v1.5.4 and above, please execute the commands below in a terminal.
    - cp ../../specs/v1.5/swagger-saral-maintenance.yaml ./src
    - cp ../../specs/v1.5/swagger-saral-frontend.yaml ./src
    - cp ../../specs/v1.5/swagger-saral-apidoc.yaml ./src
- Install npm packages by running “npm i"
- Create a subfolder inside the backend folder name it as “config.”
- Make sure Mongodb database is installed and is running.
- Now create a file in **config\dev.env** and paste the below commands
    - PORT=3000
    - MONGODB_URL=mongodb://localhost:27017/saraldata_dev
    - JWT_SECRET=SARALDATA_NODE
- Run “npm run dev” command from a terminal.
- Successful backend local deployment should show Server is up on port 3000

Deploying saral backend as Docker container:

- Open Terminal and clone source code “git clone https://github.com/Sunbird-Saral/Project-Saral.git.”
- Check latest release https://github.com/Sunbird-Saral/Project-Saral/releases.
- Change Directory to Project-Saral/ folder and switch to release tag as per release notes. git checkout tags/<tag_name>
- Go to folder path “Project-Saral/v1.0/backend”.
- From Saral release v1.5.4 and above, please execute the commands below in a terminal.

    - cp ../../specs/v1.5/swagger-saral-maintenance.yaml ./src
    - cp ../../specs/v1.5/swagger-saral-frontend.yaml ./src
    - cp ../../specs/v1.5/swagger-saral-apidoc.yaml ./src
- Dockerfile is available under "Project-Saral/v1.0/backend" folder.
- Build docker image using docker build . -t saral-backend:1.0-latest the command from "Project-Saral/v1.0/backend" folder in a Terminal.
- docker run --name saral-backend -p 3000:3000 -it saral-backend:1.0-latest

  or
- docker run --env PROFILE=dev --env PORT=3005 --env MONGODB_URL=mongodb://localhost:27017/saralnew --env JWT_SECRET=SARALDATA_NODE --name saral-backend -p 3005:3005 -it saral-backend:1.0-latest
- Docker swarm stack reference file available at Project-Saral/v1.0/backend /saralbackend-stack.yml
- Docker-compose reference file available at Project-Saral/v1.0/backend /docker-compose.yml
- For more details refer to Project-Saral/v1.0/backend/README.md

Note: If your node server and mongodb are running on the same machine. Then accessing local mongodb from the backend running as docker container needs extra steps as follows.

1. Add Docker bridge gateway IP to IP in the MongoDB config file /etc/mongod.conf under bindIp in the network interface section.

```
# network interfaces
net:
  port: 27017

  bindIp: 127.0.0.1,172.17.0.1
```

2. Choose hostname to access mongodb server and rerun docker container with below command

```
docker run --add-host=mongoservice:172.17.0.1
 --env PROFILE=dev --env PORT=3005 --env MONGODB_URL=mongodb://mongoservice:27017/saralnew
--env JWT_SECRET=SARALDATA_NODE --name saral-backend -p 3005:3005 -it saral-backend:1.0-latest
```

Data Ingestion:

For Local development setup:

- Open terminal of machine where you have cloned the saral github repository.
- Go to folder path "Project-Saral/v1.0/backend".
- Make sure all necessary collections/Tables data json files are in ./data folder. Refer "\Project-Saral\v1.0\backend\data" for samples.
- Below commands to be executed with in saral-backend to load above golden data.
    - **>> `node ./data/import-data.js --delete`**
    - **>> `node ./data/import-data.js --import`**

For backend deployed as docker:

- First you need to come under docker container
  * docker exec -it saral-backend bash
- Go to folder path "Project-Saral/v1.0/backend".
- Make sure all necessary collections/Tables data json files are in ./data folder. Refer "\Project-Saral\v1.0\backend\data" for samples.
- Below commands to be executed with in saral-backend to load above golden data.
    - **>> `node ./data/import-data.js --delete`**
    - **>> `node ./data/import-data.js --import`**

Scale it can support:

It can support concurrency of 50-70 users can Successfully **Login and Submit** the Marksheet for **Students** at a 250-500 ***students save per second***.

Next Steps:

- **To Test API endpoints:** saral_backend.docx
- **Common Errors:** saral_backend.docx

Production deployment (on AWS cloud):

Hardware requirements:

The following minimum configuration works for a scale as detailed below:

3,000 Users (*with peak concurrency of 500-1000 users*) can Successfully **Login and Submit** the Marksheet for 4**,50,000 Students** at a **1500 *students save per second***. Test Duration 5 Minutes with average response time of 10-20 Seconds with **100% Success rate**

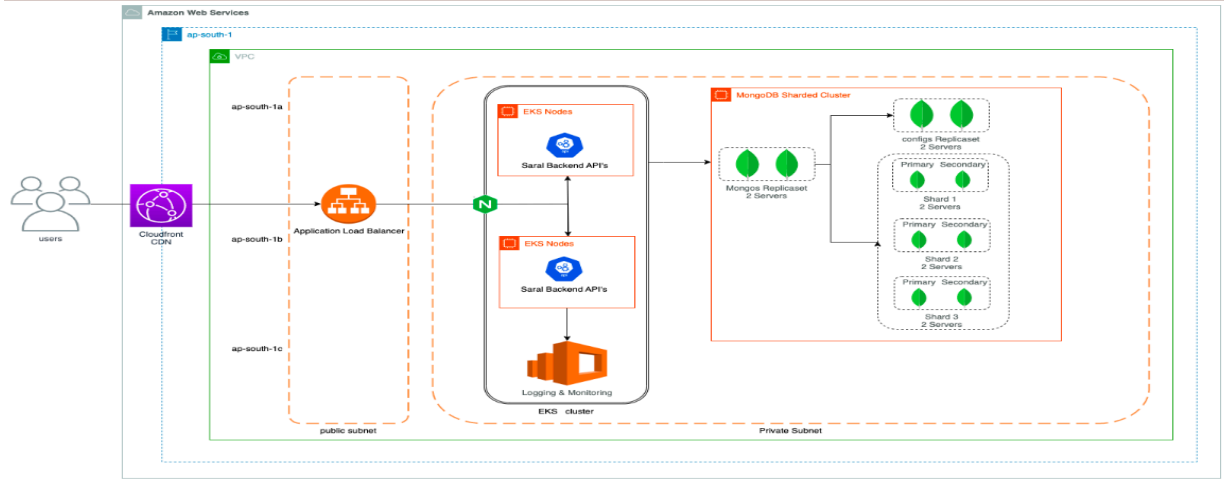
Server	Instance Type	CPU	Memory	Disk	No. Of Servers
Jenkins CI/CD	m5a.large	2	8	50	1

Monitoring and logging	m5a.xlarge	4	16	50	1
EKS nodes	c5a.xlarge	4	8	50	2
MongoDB standalone	m5a.xlarge	4	16	100	1
This is optional if you want scale or else standalone is enough MongoDB as sharded cluster(3 sharded cluster) mongoconfig	m5a.xlarge m5a.large	4 2	16 8	100 50	8 2

Network Ports requirement:

- 80 (nginx is configured in jenkins)
- 443 (nginx is configured in jenkins)
- 3005 (required to launch node server)
- 27017(to launch mongodb)
- 8080(for jenkins)
- 22(for ssh login to EC2)
- 5601(kibana dashboard for logging access)
- 30005(monitoring infra using Prometheus)
- 30001(Grafana dashboards)
- 9001(node exporter)

Deploying saral backend as Kubernetes cluster nodes:



1. Create EKS cluster along with VPC using terraform.

We use Elastic Kubernetes Service (EKS) cluster to deploy and manage a Node.js API application in a scalable and resilient containerized environment.

- First create a EC2 with machine type 1vcpu 1GiB Memory(t2.micro) linux(ubuntu 20.04) os.
- create a IAM role with admin access and assign it to this vm.
- After that login to ec2 machine and carry out below steps.
    - **install AWS CLI**
        - curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
        - unzip awscliv2.zip
        - sudo ./aws/install
    - **Install kubectl**
        - https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/
    - **Install terraform**
        - Refer: https://github.com/Sunbird-Saral/Project-Saral/tree/main
        - Note: please follow the steps mentioned in “terraform installation” file.
    - **EKS Cluster creation**
        - “ aws configure ” after that give necessary details like (give access key, secret key, default region)
        - **Note**: before that create a user in AWS IAM service and create access key and secret key and copy it and keep it safe.
    - **Run terraform template**
        - clone github repo https://github.com/Sunbird-Saral/Project-Saral/tree/main
        - cd into “terraform-templates-test" folder and execute below steps one after the other

- terraform init ---> terraform validate ---> terraform plan ---> terraform apply
- **Note**: if no errors on the screen than, EKS cluster along with VPC, NAT, IGW & routes will be created in the AWS.

2. Setup kubectl to interact with EKS cluster

Kubectl is a command-line tool that allows us to control and manage Kubernetes clusters, including EKS clusters, by sending commands to the cluster's API server.

- Install jenkins in a vm(create manually of machine type m5a.large) in public subnet also install Ansible, kubectl & AWS CLI inside the same VM.

https://github.com/Sunbird-Saral/Project-Saral/blob/main/install_jenkins.sh

- Configure the kubeconfig(EKS config) file with the jenkins using below steps.
  - Copy the kubeconfig file with t2.micro (machine where terraform was initiated) m/c to jenkins m/c.

    NOTE: Create a config file under .kube folder and copy the kubeconfig to this config file in jenkins machine and do “aws configure” before proceeding further.

  - Run the following command to update your Kubernetes configuration file with the necessary details for connecting to the EKS cluster:
    - aws eks update-kubeconfig --name <your-cluster-name> --region <your-region>

      Ex:- aws eks update-kubeconfig --name eks-cluster-saral --region ap-south-1

  - Verify that the configuration has been updated by running the following command: kubectl config get-contexts
  - Switch to the context of your EKS cluster by running the following command: kubectl config use-context <your-cluster-name>

    Ex:- kubectl config use-context saral

  - Run below commands to check your k8’s credentials

    kubectl config set-credentials cluster-admin --client-key=~/.kube/admin.key

    aws sts get-caller-identity

  - You can now interact with your EKS cluster using kubectl commands. For example, to get a list of nodes in the cluster, run the following command: kubectl get nodes

3. Create Mongodb sharded cluster.

We recommend using MongoDB sharded cluster to horizontally scale your MongoDB database, distribute data across multiple servers (shards), and ensure high availability, improved read/write performance, and the ability to handle large data volumes effectively.

Setup Sharding:

- Create the vm's manually 10vm's for MongoDB in private subnet (6vm for shard (m5a.xlarge), 2vm for mongos (m5a.xlarge) & 2vm for mongoconfig (m5a.large))

For example, this reference backend is using 3sharded mongoDb cluster in which each shard consists of 2vm's

- To install MongoDB shard cluster on 10vm's we use ansible playbooks to install this.
    - clone the repo from github https://github.com/Sunbird-Saral/Project-Saral/tree/devopstestimp2/ansible-roles and execute it so that it will install mongoDB on all the 10VM's.
    - After that set replica set for mongoconfig servers using below steps:
        - Login to mongoconfig1 m/c and do sudo vi /etc/mongod
        - Find and update replSetName : <anyname> & save the file and exit
        - After that in config1 m/c type command mongosh --host mongo-shard-configs-1 after that inside mongosh command execute below piece of code.

```
rs.initiate(
 {
  _id: "<replicaset name>",
  configsvr: true,
  members: [
   { _id : 0, host : "<IP address of mongoconfig m/c1>:27017" },
   { _id : 1, host : "<IP address of mongoconfig m/c2>:27017" }
     ]
 }
)
```

Use example shown below:---

```
rs.initiate(
 {
   _id : "saralshard1",
   members: [
    { _id : 0, host : "10.0.8.220:27017" },
    { _id : 1, host : "10.0.5.181:27017" }
   ]
 }
)
```

- After Update mongod.conf file.
- update shard name vi /etc/mongod.conf
- Add below lines

```
sharding:
   clusterRole: shardsvr
replication:
  replSetName: <replicaset name as set above>
```

Create DB and Enable sharding:

- login to mongos m/c and type command “mongosh”, than follow below steps for each collection.
  - first create database <dbname>
  - create collection --- db.createCollection("collection_name")
  - enable sharding for that DB --- sh.enableSharding("dbname")
  - create index ---- db.<collection_name>.createIndex({"keyId":"hashed"})
  - create sharding on collection ---- sh.shardCollection("<dbname>.<collection_name>", { "keyId": "hashed" })
  - check distribution ---- db.<collection_name>.getShardDistribution()
  - After this exit from mongosh

Note: Make sure to choose proper sharding key for optimum performance example:

- Marks collection sharding key is "`shardedKey`"

Now loading data into DB

- Ensure data in JSON format in loaded into the the mongos machine and import data using below command

  mongoimport --db <dbname> --collection < collection_name" > --file <path of the data file> --jsonArray

  Ex:- mongoimport --db saralv2_DB --collection classes --file /home/ubuntu/testdata/classes.json --jsonArray

4. Loadbalancer and DNS setup

We need a load balancer and DNS setup to ensure high availability, scalability, and efficient distribution of incoming network traffic. DNS translates user-friendly domain name to IP addresses allowing clients to locate and connect to required server.

- Create a domain name for Saral Backend API url and map it with load balancers.
- Configure CNAME or A record of AWS ec2 / Azure VM / Oracle VM to the domain name
- Create a SSL certificate for the domain name.

Jenkins Build & Deployment(CI/CD setup)

Jenkins CI/CD (Continuous Integration/Continuous Deployment) is needed to automate and streamline the software development lifecycle, enabling frequent code integration, automated testing, and automated deployment. This helps ensure code quality, accelerate software delivery, and enhance collaboration among development and operations teams.

1. Install jenkins on a vm in public subnet by using shell script mentioned in the github repo

   https://github.com/Sunbird-Saral/Project-Saral/blob/devopstestimp2/install_jenkins.sh

2. update docker credentials in credentials store in jenkins.

3. After that create a new job in jenkins for build and deploy to EKS cluster by using the jenkinsfile in the github repo by the file name "Jenkinsfile-prod"

Saral backend setup minimum AWS Cost Estimate:

SYSTEM CONFIGURATIONS								
Server	Instance Type	CPU	Memory	Disk	No. Of Servers	Cost per month	Utilization	1 Year r
Jenkins CI/CD	m5a.large	2	8	50	1	$48	on demand 100%	
Monitoring and logging	m5a.xlarge	4	16	50	1	$86		
EKS nodes	c5a.xlarge	4	8	50	2	$147		
EKS(Managed service)						$73		
MongoDB standalone	m5a.xlarge	4	16	100	1	$91		
						$445		$

Other Service	Pricing
Virtual Private Cloud Cost	
2xNAT Gateway for Public Subnet with 100GB of data transfer = $92.96	
Inbound Data Transfer = 1 TB = $0	
Outbound Internet Data Transfer = 100GB = $12.93	

Next Steps:

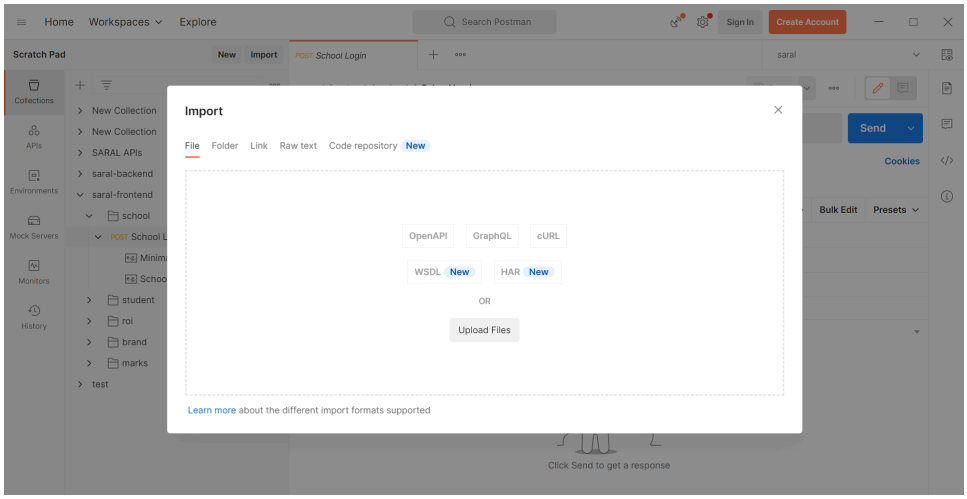
- **To Test API endpoints:** saral_backend.docx
- **Common Errors:** saral_backend.docx
- **Monitoring Infrastructure:** saral_backend.docx

Data Processing/Dev testing:

Note: Before this step make sure to add your nodejs server ip address/domain to whitelist array either in config file i.e, for example add "WHITELIST_DOMAINS=['https://saral-dev-api.anuvaad.org', 'https://saral-api.anuvaad.org', 'https://test-api.dummy.org'']" in dev.env

or

inside file Project-Saral\v1.0\backend\src\app.js add to variable named "`whitelist`"

- Download the postman application and import the collections from json files from path SARAL POSTMAN COLLECTION LINK.
- Select the import option in postman to import the collection. Please refer to the screenshot
- 

- Make sure to set
    - proper http methods: - GET, POST, PATCH, PUT, DELETE
    - proper url
    - proper header e.g
        - **Authorization** - Bearer Token
        - **methods -** http methods
        - **Origin –** url

Common pitfalls:

while setup:

- Version Conflicts: Using incompatible versions of Node.js and npm packages.

Solution: Use a version management tool like NVM for Node.js and npm's package-lock.json to ensure consistent versions.

- Missing Dependencies: Forgetting to install project dependencies.

Solution: Run npm install to install required packages listed in package.json.

- Environmental Variables: Not setting required environment variables
- syntax errors or logical errors

Mongodb database setup:

- No Indexing: not using necessary indexes or over indexing setup can cause performance degradation
- Not Handling Connection Errors: Not handling MongoDB connection errors gracefully can crash your Node.js application
- Ignoring Connection Pooling: Creating a new database connection for each request can be inefficient and lead to resource exhaustion.
- Neglecting Performance Optimization: Not optimizing queries, using appropriate indexes, or considering sharding can lead to poor database performance.

Solution: Continuously monitor and analyze your MongoDB performance, use the explain method to optimize queries, and scale your database using sharding when necessary.

Monitoring backend infrastructure

install prometheus and grafana for monitoring.

1. install helm on the jenkins machine using below commands

- curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null
- sudo apt-get install apt-transport-https –yes

- echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list
- sudo apt-get update
- sudo apt-get install helm

2. After that take the prometheus-values.yaml separately so that we can add the details to scrap metrics i.e node expoter

- helm inspect values <repo>/<chartname> > prometheus-values.yaml

ex:- helm inspect values prometheus-community/kube-prometheus-stack > prometheus-values.yaml

3. Now open the prometheus-values.yaml and the details of machine ip address and port number and save the file.

ex:- - job_name: "mongo1"

static_configs:

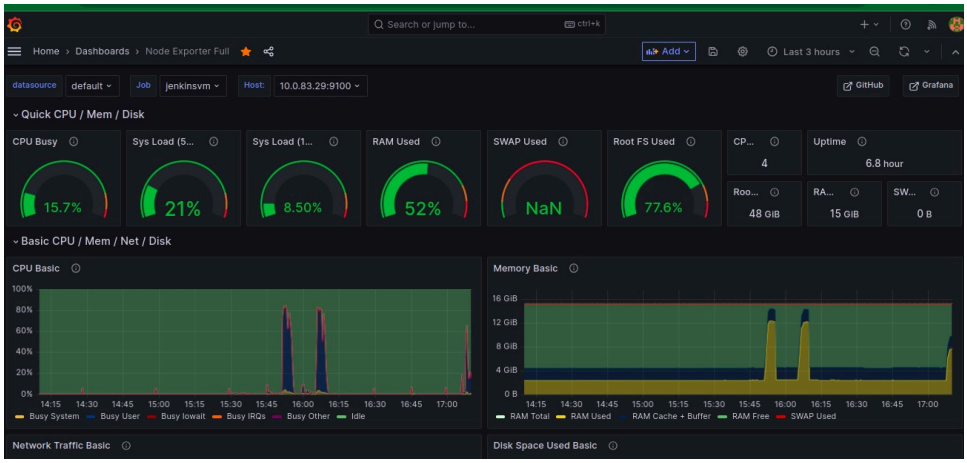
- targets: ["10.0.8.220:9100"]

4. Now by using below command you can deploy monitoring stack.

- helm install <release name> <repo>/<chartname> -n <namespace> -f prometheus-values.yaml

ex:- helm install prometheus-stack prometheus-community/kube-prometheus-stack -n monitoring -f prometheus-values.yaml

5. Now you can portforward to jenkins machine and access the prometheus and grafana dashboard.



Update/Upgrading backend:

- Open Terminal and clone source code “git clone https://github.com/Sunbird-Saral/Project-Saral.git.”
- Checkout to “main” branch “git checkout main”
- Make sure you have the latest changes. Do “git pull”.
- Create a new feature/bugfix branch “git checkout -b ＜new-branch＞”
- Make necessary code changes
- **Build:** If backend server is already setup for local dev and is running. It will auto refresh or else run “npm run dev” command from a terminal.
- **Test:**
    - Make sure to add necessary test cases to get 85% coverage. To check run “npm run test”
    - Follow “Data Processing/Dev testing” section to test API endpoints.
- **Commit:** Use below commands to push code to github
    - git add .
    - git commit -m “some message”
    - git push
- **Merge:**
    - Raise a pull request on github from your branch to “v1-develop”. Add required reviewers.
    - Upon approval and build testing for expected functionality. Merge it

Generic steps to saral backend setup:

1. **Prepare necessary hardware requirement for development environment**
    a. Sample Hardware configuration snapshot
        i. Any OS
        ii. 16 GB of System RAM (minimum requirement)
        iii. 4 core CPU (minimum requirement)
        iv. 250GB HDD
2. **Clone the project repo**
    a. Open Terminal and clone source code “git clone https://github.com/Sunbird-Saral/Project-Saral.git.”
3. **Install necessary dependencies**

a. The above cloned repo uses nodejs and mongodb technologies. If you wish to continue with the same please follow installation steps provided in previous steps.
b. If you wish to use a different tech stack like Java instead of NodeJs and mysql instead of mongodb. Please use above repo code as reference and build REST APIs by referring to postman collection (https://github.com/Sunbird-Saral/Project-Saral/blob/v1-develop/v1.0/backend/test) for endpoint details.

4. **Database Setup/Data Ingestion**
    a. The reference saral backend uses mongodb as Database, but you can use any database with necessary schemas. To know different schemas/Tables Refer path Project-Saral\v1.0\backend\src\models
    b. Find sample/Initial data to be loaded under path Project-Saral\v1.0\backend\data
5. **Data Processing**
    a. You can use any API testing tool like POSTMAN to test API endpoints. To know about sample payload and responses refer https://github.com/Sunbird-Saral/Project-Saral/blob/v1-develop/v1.0/backend/test