Supply Chain Management System

Version 1.0

Team Members – Group G

Sravanthi Nuthula

Kunal Kapoor

Vaishnavi Pentaparthi

Arjun Rehal

1. Document Overview

This document defines the high-level requirements for Supply Chain Management System. This document will be used as the basis for the following activities:

- Creating application design
- Developing and testing the application
- Analyse potential issues / bottle necks in the system and counter planning for the same.
- Determining project success

2. Project Overview

The primary aim of this project is to design and utilize a web application that would enable Company owners to procure and deliver goods and help them automate demand planning and application should enable consumers to purchase/access goods or services. The project wishes to achieve the following objectives:

- Creation of a useable database to store goods and services the company wants to outsource from given list of suppliers. Store the end products and consumers in the database.
- Developing an API which allows admin to access the supplier and products info. Enables customer to order products.
- Analysing bottlenecks and load balancing of the application.
- The tool must be maintained and kept up to date, user-friendly, transportable, and easily accessible by multiple users.

3. Product Requirements

Application Users:

Those who will primarily benefit from the new system and those who will be using the new system include:

Admin: who has control over the application – can manage suppliers and products.
Supplier: Seller - Who delivers stock to the company and track their stock order
Customer: Buyer - Who can login into the system and browse through all the products and should be able to place and track order.

Requirement ID	Requirement Statement	Must / Want
1 - Admin	Admin should be able to login into the system Able to add and manage different products Able to see different suppliers and stock information from suppliers	Must

	Admin should be able to add and manage new suppliers Admin should be abe to see all products and current/pending customer orders and stock orders	
2 - Supplier	Supplier should be able to login Supplier should be able to check the stock order status	Must
3 - Customer	Customer should be able to register and login Able to see different products Able access particular product details Able to track the customer order status	Must

4. Use Cases

We are targeting to implement only priority HIGH use cases in this milestone 2

Public endpoints

/register - /login - /getproducts – /getProductDetails

Use Case ID	**1**
Use Case Name	Register a Customer
End point & type	/registerCustomer -POST – validUser JSON object
Description	Given all the information, used should be able to register into the application
Priority	High
Frequency of Use	Multiple per app load
OnSuccesss	Message - "successfully registered as a customer"
OnFailure	Message - Invalid details
Request Body	{ name : Sravanthi, email: sravanthi@gmail.com, phoneNumber : ,postalcode : }

Use Case ID	2
Use case Name	User should be able to login
End Point & Params	/login – GET - valid username(email) and password
Description	Able to login by giving valid username and password

Priority	High
Frequency of Use	Multiple per app load

Response

OnSuccess	Message - successfully logged in
OnFailure	Message - Invalid credentails
Request Body	null
Request Params	Username , password
Response body	{ name : Sravanthi, email: sravanthi@gmail.com, phoneNumber : ,postalcode : }

Use Case ID	3
Use case Name	Get all products from the database
End point & type	/getProducts – GET - No params required
Description	Get all products from the database irrespective of customer authority
Priority	High
Frequency of Use	Multiple per app load

Response

OnSuccess	Message - products found
OnFailure	Message - – No products found
Request Body	null
Request Params	null
Response body	[{productId:123,productName : shampoo ,category: bodycare }, {productId:124,productName :conditioner,category: bodycare }]

Use Case ID	4
Use case Name	Get product details based on customer request
End point & type	/getProductDetails : GET
Description	Given valid productID customer should be able to see the all product details
Priority	High
Frequency of Use	Multiple per app load

Response

OnSuccess	Message - product found
OnFailure	Message - – No product found
Request Body	null
Request Params	ProductId
Response body	{productId:123,productName : shampoo ,category: bodycare , stockQty : 25, unitPrice : $10}

Internal endpoints

Customer Orde: /placeCustomerOrdeR - /trackCustomerOrder

Use Case ID	5
Use case name	Customer should be able to place an order
End point & Params	/placeCustomerOrder : POST - CustomerOrder JSON Object
Description	Given valid information of the product and customer, order should be successfully placed
Priority	High
Frequency of Use	Few per app load

Response

OnSuccess	Message - successfully placed the order
OnFailure	Message - Unable to process the request
Request Body	{Useid:123,productID :345,category: bodycare , orderQty : 25,orderAmount: $10}
Request Params	null
Response body	customerOrderID

Use Case ID	6
Use case name	Customer able track the order given customer order ID
End point & Params	/trackCustomerOrder : GET - Valid customerOrderID
Description	Able to see status of the order
Post conditions	User should be able to track order
Priority	High
Frequency of Use	Few per app load

Response

OnSuccess	Message - successfully placed the order
OnFailure	Message - Unable to process the request
Request Body	null
Request Params	customerOrderID
Response body	{status : “in progress”,customerOrderID: 789,userid:123,productID :345,category: bodycare , orderQty : 25,orderAmount: $10}

Stock Order and Stock: /getStockInfo - /placeStockOrder - /trackStockOrder

Use case ID	7
Use case name	Admin and supplier should be able to track the stock order given valid stockOrderID
End point & Params	/trackStockOrder : GET - Valid stockOrderID
Description	Can see the status of the stock order
Priority	High
Frequency of use	Few per app load

Response

OnSuccess	Message - got the status of the order
OnFailure	Message - Unable to process the request

Request Body	null
Request Params	stockOrderID
Response body	{status : “in progress”,stockOrderID: 789,userId:123,productID :345,category: bodycare , orderQty : 25,orderAmount: $10}

Use case ID	8
Use case name	Admin should be able to place stock order
End point & Params	/placeStockOrder - POST: Valid stockOrder JSON Object
Description	Admin should be able to place stock order given valid supplier and stock info
Post conditions	User should be able login into the system
Priority	High
Frequency of use	Few per app load

Response

OnSuccess	Message - successfully placed the stock order
OnFailure	Message - Unable to process the request
Request Body	{userID:123,productID :345,category: bodycare , orderQty : 25,orderAmount: $10}
Request Params	stockOrderID
Response body	{status : “in progress”,stockOrderID: 789,userId:123,productID :345,category: bodycare , orderQty : 25,orderAmount: $10}

Use case ID	9
Use case name	Admin should be able to see all the stock information
End point & Params	/getStockInfo : GET - Valid StockID/ ProductID
Description	Admin should be able to view stock information of all the products
Post conditions	User should be able login into the system
Priority	Medium
Frequency of use	Few per app load

Response

OnSuccess	Message - successfully retrieved the stock
OnFailure	Message - Unable to process the request
Request Body	null
Request Params	stockID / productId
Response body	{stockId : 124, userId:123,productID :345,category: bodycare , stockQty : 25, unitPrice : $10}

Product CRUD - /addProduct - /updateProduct - /deleteProduct

Use case ID	10
Use case name	Admin should be able to add new Product given valid product info
End point and params	/addProduct – Product JSON Object
Description	Product should be added to database
Priority	High
Frequency of use	Few per app load

Use case ID	11
Use case name	Admin should be able to update the existing Product info given valid product info
End point and params	/ updateProduct – Product JSON Object
Description	Product should be updated to database
Priority	Medium
Frequency of use	Few per app load

Use case ID	12
Use case name	Admin should be able to delete the product given valid productID
End point and params	/deleteProduct – valid ProductID
Description	Product should be deleted from the database
Priority	Medium
Frequency of use	Few per app load

We will be adding One supplier in the database directly in this milestone2 and all CRUD operations may not be available.

Supplier CRUD : /getSupplierDetails - /getSuppliers – /addSupplier - /updateSupplier – /deleteSupplier

Use case ID	13
Use case name	Admin should be able to see all the info of a supplier
End point and params	/getSupplierDetails – valid UserID
Description	Supplier info should be loaded from the database
Priority	Medium
Frequency of use	Few per app load

Use case ID	14
Use case name	Admin should be able to see all the suppliers
End point and params	/getSuppliers– No params – only admin can access
Description	All the suppliers should be loaded from database
Priority	High
Frequency of use	Multiple per app load

Use case ID	15
Use case name	Admin should be able to add the new supplier
End point and params	/addSupplier – User JSON Object
Description	Supplier should be added into the Database
Priority	Medium
Frequency of use	Few per app load

Use case ID	16
Use case name	Admin should be able to update the existing supplier info
End point and params	/updateSupplier – User JSON Oject
Description	Supplier data should be updated into the database.
Priority	Medium
Frequency of use	Few per app load

Use case ID	17
Use case name	Admin should be able to delete the existing supplier
End point and params	/deleteSupplier– valid UserID
Description	supplier should be deleted from database
Priority	Medium
Frequency of use	Few per app load

5. Tech Stack

We will be using MERN stack to develop the application. As for as the milestone 1 is concerned we are targeting to develop a backend node API which implements endpoints for the functionalities of the application.

We are targeting 2 layers (Application Layer and Database Layer) of 3 tier architectures.

MERN - MongoDB   Express React Node and Git for Version Control

Database :

WHAT: As for as the Database Layer is concerned, we are going to use MongoDB Atlas which is NoSQL Document Model Database to store and retrieve our data.

WHY: Mongo DB suitable for initial development process, here we are trying to develop Minimum Viable Product (MVP) so Mongo is the best solution as we do not have to spend much time on designing the model.

Mongo DB Atlas – we created account in mongoDB atlas and singed up for free version to store and retrieve the data via Google Cloud Platform. We have choose free cluster from MongoDB Atlas.

M0 (Free Cluster) – uses MongoDB 5.0 version

Node JS :

Node js is used for both front end and backend development. It is a non-blocking and event-driven server. It is very fast as it is being build on chrome's V8 Javascript engine

RBAC : Using Role Based Access Control to provide security to the application and avoiding unauthorized access. Making sure only authorized person can access relevant data of the enterprise.

Application configuration :

Application will be running on Local browser on port 3000 and can be accessible by postman.

We are not using any other 3rd parties like google API. So, we don't have any authentication tokens in the configuration.
We will be configuring URL to access mongoDB.
We will create new environment variables for username and password of mongoDB

Code Review :

We will be using git to version control the application changes and team members will be making sure the changes are correct before committing them to the main branches.

One of the team members will review the code and then only author can push the code to main branch and test it on live. Person who is reviewing will make sure that no other code is broken because of new changes and performs regression testing.
It is also individual developer responsibility to test the application after every changes.

Testing :

Non-functional requirement.

Every developer will unit test the application based on their own changes.

As team everybody takes latest commit of all the developers and test the application as a whole which is termed as integration testing.

We will test the application based on requirement document and see whether it is acceptable for the client.

Technical debt:

We will make sure we are not leaving any tasks or requirement pending before delivering the project.

Deployment environments:

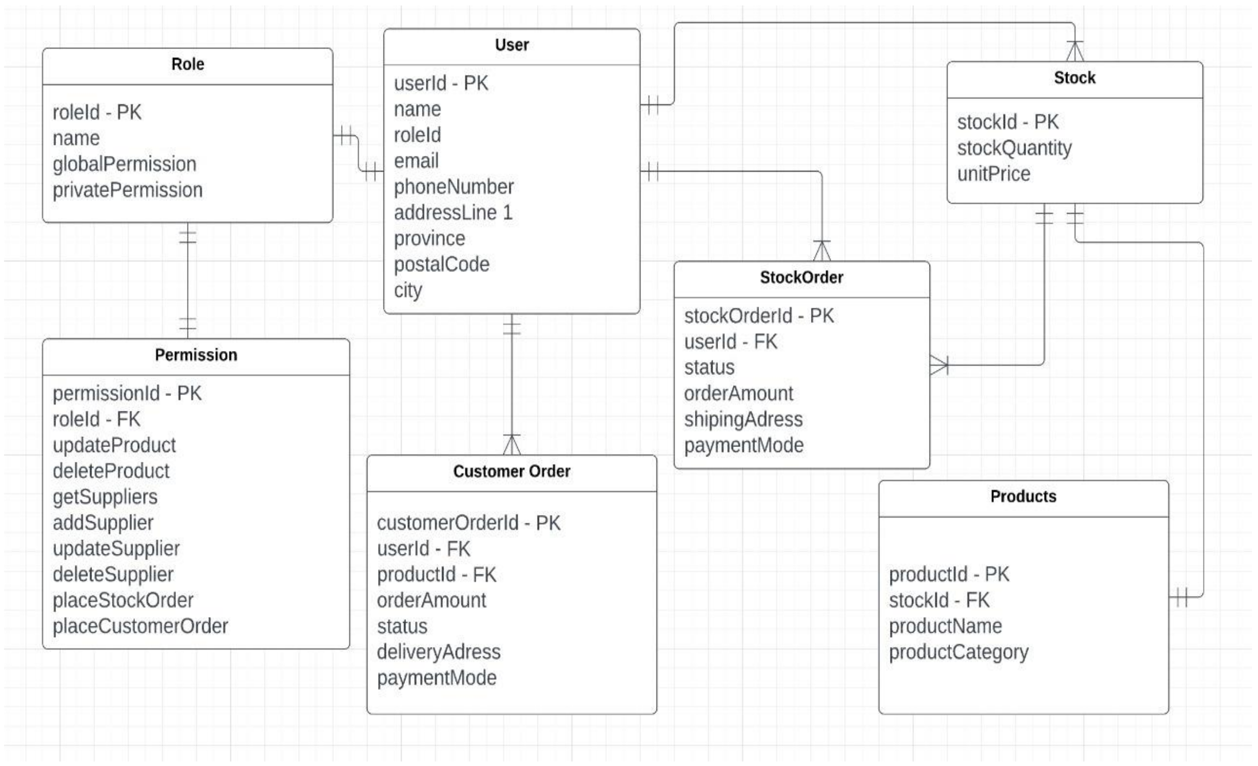
We will be creating multiple branches in the Git.

Individual developer branches: where team members can implement the code and unit test their tasks and commit them to staging

Master Branch: It contains the code for staging environment and everything will be tested here before moving it to production.

Main Branch: This will be our production branch where code can be merged from master branch (staging env)

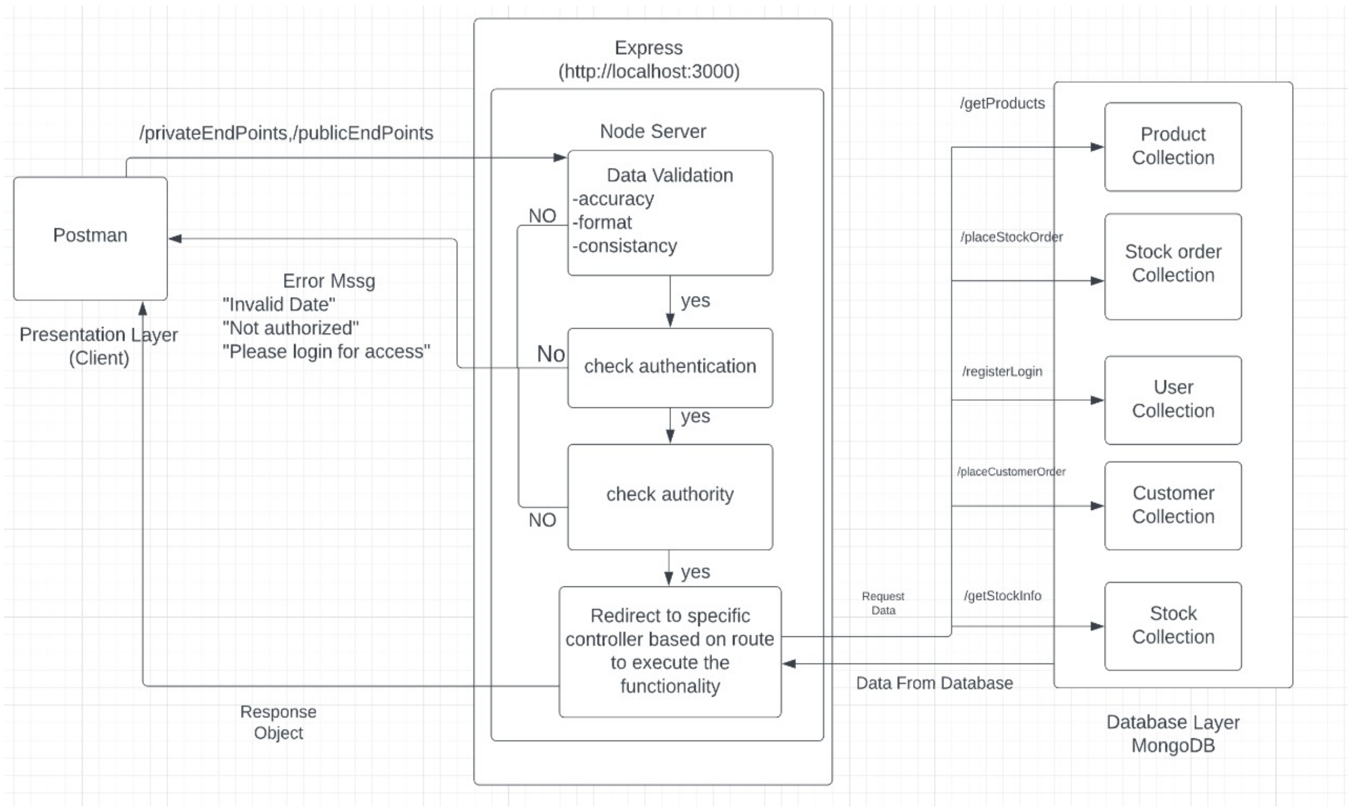
6. Database Entities and ER Diagram



*PK - Primary Key, *FK – Foreign Key

7. High Level System Design

Data validation block performs: contextual accuracy of values and consistency and format of the values



8. Bottle necks & Redundancy – Solutions

Clustering: what if the database goes down:

As we are using mongoDB Atlas it provides a cluster which means it will have multiple servers connecting to single database which will have same access to shared storage. So, if the instance that we are using goes down for a moment then another instance from the cluster of instances will serve as a database.

How much data and bandwidth is required for the system:

Bandwidth means the capacity at which a network can transmit data - **Network In/Out: 10 GB in/out.**

In seven days period mongo DB M0 cluster will allow 10GB in and out which means total data transferred into or out of the cluster should not be more than 10GB. If so, users will not be able to access database which means no create, update, delete or retrieval operations can happen on the database until seven days period completes.

Storage Capacity of the system:

Since we are using mongo DB **M0 clusters** it will allow the data upto **512 MB**. So out cluster can't be larger than 512 MB. If at all we need more storage, we need to upgrade the plan.

Multiple Connections at a time:

M0 cluster has **500 connection** limit which means at a given point of time 500 connections can be created to mongoDB which internal means 500 different users can access the database at the same time.

Indexing in the database & Increase the response time:

Considering the database is having thousands of users and based on the storage capacity of the database response time becomes lower to retrieve one particular entry from the database. Thus, query retrieval performance reduces and it makes user to get frustrated to wait until he receives response from the server.

Solution: Indexing – It speeds up the performance of the database. It used to quickly locate the data in the table without having to search every single row in the table every time when a table is accessed.

We will index the unique keys in the table so that it becomes quick while retrieving the database.

9. Maintenance, Reliability and Scalability of the application

Security & **Reliability** : RBAC make sure the we are giving access to authorized users and we authenticate every user request before passing it to controller to make sure user is registered with application.

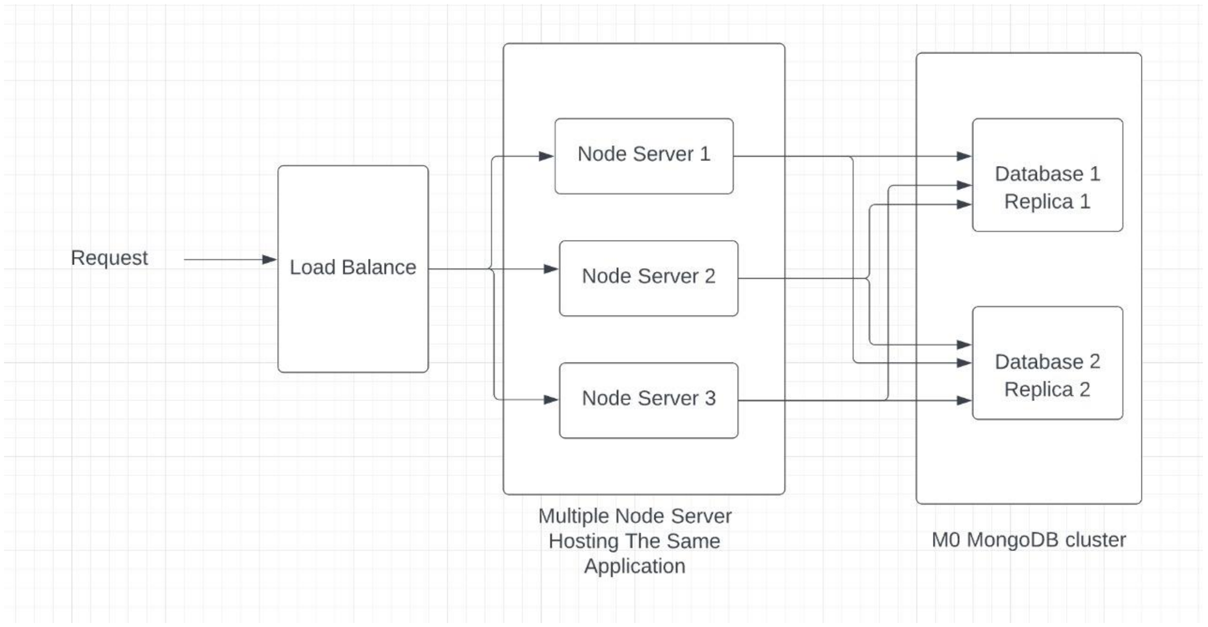
We will increase cluster size and **upgrade to new version** when user traffic increases

MO cluster – 10GB of bandwidth and 512MB storage

M2: 20GB and M5: 50 GB

If number of users increase will increase storage size. Upgrade to high cluster versions.

Load Balancing:



We have plotted above diagram to explain load balancing. If there is lot of traffic in the application then we will host out API on multiple node servers which serves the request based on availability of the resources

M0 cluster – will have 3 replica sets – If one goes down other will be up for serving. We can maintain replica of the original database on multiple instances and balance the load on different servers based on user traffic.

We are going to analyse the performance of the application using the following sequences

Sequence 1:

Who is this system for?
This system is for companies who want to manage their supply chain process.

How will this end user get access to your service?
End user can access the application through postman and sent request to specific route in particular format and receives the response.

How many users do you estimate will be on this service hourly, daily, annually?
We are expecting there will be hundreds of users daily, few double digit users hourly and millions of users yearly.

What do the requests to the system look like?
User can access get, post, put or delete requests.
Passing url in the postman: ex : https://localhost:3000/login
Request Body: user has to send username and password in the body of the request

What will users expect this service to output?

Response: Given proper credentials user will be seeing successful login message and then he can access rest of the information based on his role.
If user is hitting for https://localhost:3000/getProductDetails given productD in the request body. User will be able to see product details response.

Will system be more read heavy or write intensive?
It depends on the Role: For Admin it will be write-intensive as he is the one responsible for creating suppliers and adding new products into the system. Admin can also read the data but writing operations will be more than retrieving the data.
If the user is a customer then retrieval operations will be more as user searched for different products and go through details of individual product it will be more read-intensive.

How fast should search results be?
If the user is requesting for one particular product details then postman gives response in 1 second. It also depends on M0 cluster transmission limit on the mongDB in 7 days we can use 10GB data transmission. Speed gets effected with the increased transmission data.

Should user requests be handled in real time?
User requests will be handled in real-time Being said that M0 Cluster can accepts 500 different instances of the database running at a time. At given time at least 500 different users will be served. 501th user request gets executed when one of 500 existing instances get free. As in when traffic increases we will be upgrading the cluster version from mongo atlas.

Will web traffic always be equally distributed across all locations your service will be deployed?

As we are using load balancer, we will try to map the request to nearest server if not it equally distributes the load among all servers available.

Sequence 2:

Please refer the High Level System Design

Sequence 3:

Estimate the required storage and bandwidth ?
With M0 Cluster – we can **512MB** data in the database and **10 GB in/out transmission data** which means in **7 day period** we can hit mongodb for data worth of not more than 10GB for both read and write operations. As the storage limit is only 512MB is is very unlikely exceed 10GB writing transmission data.

How much data would you need to store individual records? e.g say 1 Tweet = 10KB
The maximum BSON document size is 16 megabytes. So our document should be less than 16MB this will make sure single document is not using excessive RAM and lot of bandwidth during transmission. One customer information document is approximately **20KB** depending how lengthy names user is filling in the form.
Object.bsonsize(doc) – gives thesize of document

How much data is this per month for your estimated number of users?
We assumed that there will be hundred users per day – then 3000 users per month.
Getting customer details while login - 40Kb (including login request and transmission bandwidth)

Accessing few product details – can be upto 100Kb if user hitting multiple times.

Per 1 user – per day – 400Kb - Per month - 1200kb – 1MB
For 3000 users – 3000 MB of data.

How many read/write requests per second are there?

Considering 3000 users per month – assuming all are customers except one admin
Customers – considering user hits 40 different products on an average, 5 requests for order tracking and some miscellaneousrequest.

One customer per day : 50 read requests – 15,000 read request per month.
Admin : 100 read requests and 50 write requests : 5000 wirte request per month.
Per second: we may expect - 10 read and 2 write request from all users.

If a popular account shares an activity, how does this impact the number of reads?

If accounts shares activity, then number of reads definitely gets increased

Sequence 4:

Design core components like the API, data model
Designed system design which includes structure of API and ER diagram to understand how data models are analogous the tables in schema.

Do your use cases point you towards a relationship data model or document-based model?
Our use cases are more flexible with document-based-model

Will you use RPC, SOAP, REST or another API model?
We will use REST API model - Represeantaional State Transfer Application Programming Interface it follows REST architectural style and provides a great deal of flexibility
Data is not tied to methods or resources, so REST can handle multiple types of requests and return different from of data.

What will the API request body/parameters contain?
API request body parameters : in canse of login user has to send
Password and user name in the body section : "username" : "Sravanthi" , "password" : "abcd123"

In case of registering a customer – customer JSON object

```
{
 "firstname" : "Sravanthi",
"lastname"  : "nuthula",
  "address" : "Toronto"
}
```

If tasks take too long, how do you store them so that their status can be later queried by a user?

As we are using node.js it will handle parallel request for us. It waits until the response is received for task then callback will be executed.

How will service authenticate and authorize users?
Authentication: we will authenticate user when logged in. If user is not logged in then access will not be given to respective funationalities
Authorization: we can Oauth 2.0 – open authorization – when user is having access token then we can consider that the user is authorized for the respective route
We are using **RBAC**: which give authority to respective functionalities based on user role.

Will any processes in your service depend on an external service?

We are not using any third party services except a mongo cluster from mongo atlas. Our database and cluster is maintained by atlas team and they will make sure about keeping database up always.

Sequence 5:
What portion of the code/system takes too long?

In general, logistic code will take longer time, which means to know the status of the order as it depends on multiple criteria including the status of delivery boy and stock in the warehouse and certain weather conditions. It has complex to guess the time of delivery.

Can certain tasks be broken up and handled by smaller child processes?
Yes, As the node.js is single threaded and it takes longer time for CPU intensive tasks, we can break such tasks into smaller child process so that one single process will not block entire event loop for a long time. And take advantage of multiple process concept.
We can easily spin a child process using node's "child_process" module. And child processes communicate within inbuild message system

Is it possible to store static files separately and closer to your users with a CDN?
Yes, we can store the static files in Content Delivery Network closer to the targeted user locations so that increase the speed of access for the user as CDN will be distributed.

How do you ensure that if a server or database goes down, your service won't go down as well?
As we are using cluster from mongo Atlas. Atlas will take care of our cluster in case one node goes down then there will be another node in the cluster which serves the purpose.

Is there a way to monitor and log the behavior of your service?

We are planning to incorporate logging services when every request hits the service. will log time and parameters and from whom the request got and location of the client. eventually will catch all exceptions in the controllers and log the exception in logger file so that developer can refer and fix the issues later. We will also capture the response in the same log file.