Supply Chain Management System

Version 1.0

Table of Contents

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 (functional and non-functional)

Requirement ID	Requirement Statement	Must/Want	Comments
1	Admin should be able to login into the system	Must	

Covers all use cases that have been chosen for implementation

Internal endpoints

/adminMenu

/addProduct -

/addSupplier

Public endpoints

/Register

/login

/home

4. System Users

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

INCLUDE Exaplanation and functionalities of the roles

- ***Customer:***

  Explanation

- ***Supplier:***

  . Explanation

- ***Admin:***

  Explanation.

- ***Product:***

  Explanation.

Functi	Customer	Supplier	Admin
	Manage own profile	Manage own profile	Manage users (add -edit -delete)
	Search for products	Outsourcing goods	Manage suppliers (add -edit -delete)

	Order products	View supplier information	Manage product categories (add -edit - delete)

Table 1: System User Functions

5. Glossary of Terms

INCLUDE new terms as the application develops

	Term	Definition
1)	Logistics	Delivery of product from orgin to consumer

6. Use Cases

Identify all use cases and prioritize those

6.1. Use Case Narrative

Use Case ID	**1**
Use Case Name	Register a customer
Created By	
Date Created	
Description	
Postconditions	User should be able to register
Normal Course	1. User opens app 2. User register themselves
Priority	High
Frequency of Use	Multiple per app load

7. Tech Stack

INCL add description under each of the sub task

MERN Stack - Mongo Express React Node

Database – Which ? Why ? Pros and cons

Application configuration

- The port to run the software on
- The authentication tokens to access other software
- The location of keys and trusted certificates (to validate client requests)
- The URL to access the database
- Environment variables

Code Review

- The purpose of setting up a code review system on a team is to ensure that the agreed quality standards are met.
- It also ensures that existing features are not broken and new bugs are not introduced.
- This process is manual.

Testing

unit, integration, system, acceptance testing

Technical debt

- Abrupt introduction of requirements leads developers to decide between getting the task done quickly with low quality or to maintain a high quality and possibly miss the deadline.
- The cost of taking the easier path, which would need to be repaid in the future, is called technical debt.

Deployment environments

Development, testing, staging, production

8. Database Entities and ER Diagram

Data models include attributes that meet the use cases and can adapt to changes

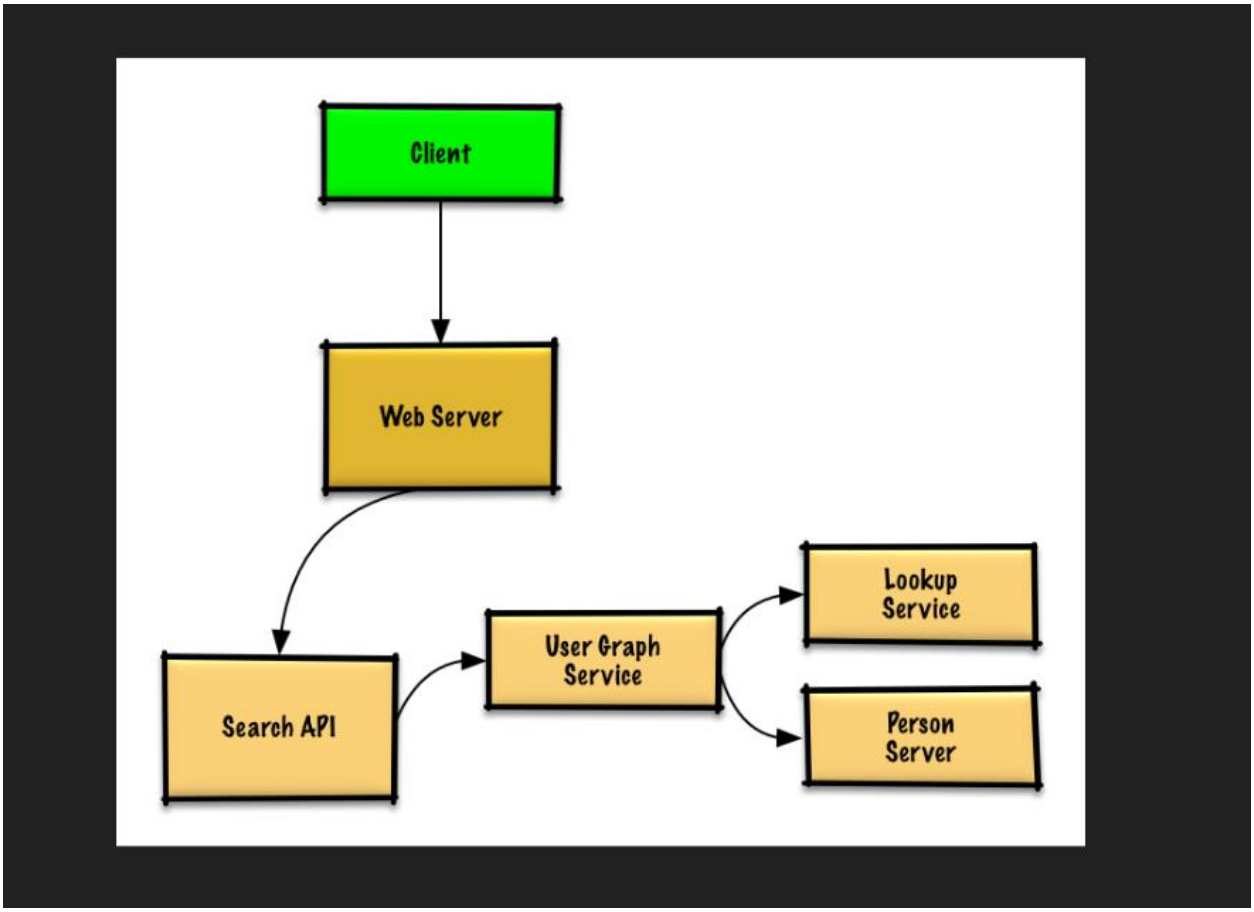
1- User: Customer who register into the application

Entity	Attribute	Description	Data Type
user	user_id	Unique Identifier	Auto increment
	email_address	email address of the user	String – Unique (Not Null)
	phone_number	user's primary phone number	String – Unique (Not Null)
	first_name	first name of the user	String
	last_name	last name of the user	String
	created	the date the record was created	Date
	last_updated	the date the record was last updated	Date

INCLUDE FIGURE OF ER DIAGRAM

9. High Level System Design

Fully displays how the backend will handle and delegate requests at scale


Client
Web Server
Search API
User Graph Service
Lookup Service
Person Server

10. API Design

IN DETAILED DIAGRAM OF WHOLE APP – (db , api, routing , load balancer, 3rd party apps, DB replicas)

Data validation handling

The following data validation rules will be applied:

1. The contextual accuracy of values.
2. The consistency among values.
3. The allowed format of values.
4. The completeness of values.

Define end points here

/home – loads overview content of application and displays in the front end

/

/

11. Bottle necks & Redundancy – Solutions

Indexing in the database

Clustering if the database goes down

How to increase the response time

Identify system constraints and capacity

how much data and bandwidth is required for the system

Rough idea of how much bandwidth is required for the system

- How much storage does the database take to store 1000 records
-

12. maintenance, reliability and scalability of the application

Describe all the points in each sequence

Describe these points in the sequence and solutions to handle the issues and adjustments in the design

Sequence 1:

Who is this system for?

How will this end user get access to your service?
How many users do you estimate will be on this service hourly, daily, annually?
What do the requests to the system look like?
What will users expect this service to output?
Will system be more read heavy or write intensive?
How fast should search results be?
Should user requests be handled in real time?
Will web traffic always be equally distributed across all locations your service will be deployed?

Sequence 2:

High level design

Sequence 3:

- Estimate the required storage and bandwidth
- How much data would you need to store individual records? e.g say 1 Tweet = 10KB
- How much data is this per month for your estimated number of users?
- How many read/write requests per second are there?
- If a popular account shares an activity, how does this impact the number of reads?

Sequence 4:

- Design core components like the API, data model
- Do your use cases point you towards a relationship data model or document-based model?
- Will you use RPC, SOAP, REST or another API model?
- what will the API request body/parameters contain?
- If tasks take too long, how do you store them so that their status can be later queried by a user?
- How will service authenticate and authorize users?
- Will any processes in your service depend on an external service?

Sequence 5:

- What portion of the code/system takes too long?
- Can certain tasks be broken up and handled by smaller child processes?
- Is it possible to store static files separately and closer to your users with a CDN?
- How do you ensure that if a server or database goes down, your service won't go down as well?
- Is there a way to monitor and log the behavior of your service?