NOTIFY PUBLISHER

A PROJECT REPORT

Submitted by

SNEHA ROY

REG NO: TKM20MCA2039

to

The APJ Abdul Kalam Technological University

In partial fulfillment of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

Thangal Kunju Musaliar College of Engineering
Kerala

DEPARTMENT OF COMPUTER APPLICATIONS

JULY 2022

DECLARATION

I undersigned hereby declare that the project report on NOTIFY PUBLISHER, submitted for partial fulfillment of the requirements for the award of the degree of M.C.A of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of Prof. Vaheetha Salam. This submission represents my ideas in my own words and where ideas or words of others have been included; I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Kollam SNEHA ROY

18/07/2022

DEPARTMENT OF COMPUTER APPLICATION
THANGAL KUNJU MUSALIAR COLLEGE OF ENGINEERING

C E R T I F I C A T E

This is to certify that, the report entitled "**Notify Publisher**" submitted by **SNEHA ROY (TKM20MCA2039),**to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications, is a bonafide record of the project work carried out by her under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose

Internal Supervisor Head of the Department External Examiner

Tekact
www.tekact.com

Tekact Private Limited,
Balaraj's Arcade, 4th Floor,
Whitefield Main Road,
Opp Brigade Cosmopolis.
Bengaluru – 560066

TO WHOM IT MAY CONCERN

This is to certify that Ms. Sneha Roy from TKM College of Engineering, Kollam is currently undergoing the internship from **April 6th, 2022** to **August 3rd, 2022** with **Tekact Pvt Ltd**. During the tenure with us, she was punctual, hardworking, and inquisitive.

We wish her all the success in future endeavors.

Sincerely,

Place: Bengaluru

Date: 15-July-2022

SVK Pillai

Director

ACKNOWLEDGEMENT

First and foremost I thank GOD almighty and my parents for the success of this project. I owe sincere gratitude and heart full thanks to everyone who shared their precious time and knowledge for the successful completion of my project.

I am extremely grateful to **Prof. Dr. Fousia M Shamsudeen**, Head of the Department, Department of Computer Applications, for providing me with best facilities.
I would like to thank our coordinator and project guide **Prof. Vaheetha Salam** , Department of Computer Applications, who motivated me throughout the project .

I would like to thank my external coordinator **Mr. Rajesh S R**, Tekact Private Limited, who guided me throughout my work.

I profusely thank all other faculty members in the department and all other members of TKM College of Engineering, for their guidance and inspirations throughout my course of study.

I owe my thanks to my friends and all others who have directly or indirectly helped
me in the successful completion of this project.

SNEHA ROY

ABSTRACT

In event driven architecture, when a service does some piece of work that other services may be interested in, that service generates a record of the done action in event-driven architecture. Other services consume such events in order to fulfil any duties that arise as a consequence of the event. Services that produce requests, unlike REST, do not need to know the specifics of the services that consume the requests. Here's an easy example: When a customer places an order on an ecommerce site, a single "order placed" event is generated, which is subsequently received by numerous microservices. There are several methods for publicizing events. For example, they can be published to a queue, which ensures that the event is delivered to the proper consumers, or they can be published to a "pub/sub" model stream, which broadcasts the event and provides access to all interested parties. In either situation, the event is published by the producer, and the consumer gets it and reacts appropriately. It should be noted that in certain circumstances, these two players are also referred to as the publisher and the subscriber. This project provides a multi-tenant solution based on a micro-service architecture that would broadcast target-based events. It will also enable APIs for users to create Apps that can function across many domains.

Contents

1. INTRODUCTION 1

1.1 Objective …….....2

1.2 Company Profile …….....3

2 LITERATURE SURVEY 7

2.1 Purpose of literature review ……..… 7

2.2 Related works ……...8

2.2.1 MircoServices…….....8

2.2.2 Spring Boot ……....9

2.2.3 PostgresSql …….……....10

2.2.4 RestFul API …….....10

2.2.5 Swagger ……11

2.2.6 Maven ……12

2.2.7 Postman ……12

3 METHODOLOGY 14

3.1 Architectural Design ……14

3.2 Module Description …….....16

3.2.1 Admin ……16

3.2.1.a Organization …….....17

3.2.1.b Tenant …….……..17

3.2.1.c Apps …….....18

3.2.1.d Users…….....18

3.2.1.e User Groups ……...…...18

3.2.1.f Roles ……...……..19

3.2.2 Common ……...19

3.2.3 Document Service ……19

3.2.4 Master Data……20

3.2.4.a Master Data Location…….....21

3.2.4.b Master Data Type …….....21

3.3 RESTFul Webservices ……..22

3.3.1 Working ……22

3.3.2 Client Request ……...22

3.3.3 Server Response ……23

3.4 System Specifications . . .... …….....23

3.4.1 Software Specification ……..23

3.4.2 Hardware Specification ……24

3.4.3 Server Specification ……..24

3.4.4 Software Description ……24

3.5 System Design ……..30

3.5.1 Logical Design . ……...…30

3.6 Input Design ……...31

3.7 Output Design ……33

4 RESULT AND DISCUSSION **35**

4.1 Testing methods ……...35

4.1.1 Unit Testing with Mockito…….36

4.1.2 Integration Testing…….…36

4.2 Output Screens and Results……37

5 CONCLUSION **39**

5.1 Future Enhancement ...……...…...…...……..39

6 REFERENCES **40**

APPENDIX **41**

List of Figures

3.1Architectural Design……15

3.2 Module Design…….....16

3.2.1 Admin Design…….....17

3.2.4 MasterData Design……...20

3.3 REST API Architecture……...22

4.1.2 Testing Modules……...37

A.1User Controller……..…….....41

A.2 MasterData Controller……41

A.3 User Service Test Coverage……...…….....42

A.4 Master Data Service Test Coverage.....42

CHAPTER 1

INTRODUCTION

Notify Publisher is a system being developed for Tekact Private Limited to publish target-based events. This is a product based on microservices architecture where each microservice handles problems specific to a particular domain. The users will be creating apps by stitching those various microservices. This product aims to provide APIs for users to compose Apps that can work with various domains. Notify Publisher is a multi-tenant system. The whole human category including teachers, students, businessmen, housewives are having a busy hectic schedule and today's life is filled with responsibilities and stress. In between these, it's always difficult to recollect everything like the dates to resubmit or pay back your bills. In our developing and technology-dependent life we totally depend upon gadgets, especially smartphones. With this, we get a chance to use technology in an exceedingly better way in order that it may be made useful to use and it plays a crucial part in our standard of living and helps us stay slot in some ways. So, we reasonably tend to make an answer for our lives to form it simpler and easier within the of Notify Publisher by combining the matter and technology.

Nowadays many organizations are driving technology innovations and creating reusable services/APIs enables faster time to market for new products. This project, which is based on microservices, demonstrates how multiple services can run independently by utilising the most effective patterns for microservices in order to enable scale, performance, and resilience.

1.1 Objective

The project aims to achieve the following for the Notify Publisher product:

1. Provide APIs to create Apps:
   This project provides APIs that can create the Apps. Companies can make the data and functionality of their programs available to external third-party developers, business partners, and internal divisions inside their own organizations through application programming interfaces, or APIs. Through a specified interface, this enables services and products to interact with one another and use one other's data and capability. Developers merely use the interface to interact with other goods and services; they are not required to understand how an API is developed. Over the past ten years, the use of APIs has increased to the point that many of the most well-liked web services today would not be viable without them.
2. Provide ways to compose apps using such APIs:
   This project provides ways to compose apps creating Apps. An application programming interface, also known as an API, is a compiled set of defined rules that explains how various programmes or computers communicate with one another. API stands for "application programming interface." An intermediary layer called an API lies between an application and the webserver to handle data flow between the two. While the calls and responses in this procedure are all made over an API, the data transmission will vary based on the web service being used. A user interface is made for humans to use, whereas an API is made for a computer or application to use. Easy integration via the usage of REST APIs:

   Applications that make use of a REST API architecture are intended to be quickly and loosely linked. Although this architecture focuses on data transfer between clients and servers, it can also be used for user interaction and system interaction. In the world of developing apps, the demand for better, quicker, and simpler APIs is on the rise, and REST APIs are taking the lead in the discourse around legacy modernization. Because we are leveraging REST APIs, integration is made simpler.

3. Workflow management:

   This project's overarching goal is to keep the workflow running smoothly. Workflow management is the practise of creating, documenting, monitoring, and improving upon a sequence of steps in an organization's processes. Workflow, which is necessary in order to accomplish a task to one's satisfaction successfully. The goal of workflow management is to optimize the steps in the workflow in order to guarantee that the task will be completed in an accurate, consistent, and time-saving manner. This can be accomplished by reducing the amount of time spent on each step.

4. Provide a working App:

   On the request for an App, only the working Apps will be provided through API requests. If the request is for an id that is not there then it will send an error message requesting you to provide a valid App id. To obtain information, a client application launches an API request, also known as an API call. The Uniform Resource Identifier (URI) of the API is used to handle this request, which comes with a request verb, headers, and occasionally a request body, from an application to the web server. When the API receives a legitimate request, it calls the external application or web server. The server replies to the API with the data that was requested. The API sends the data to the application that made the initial request. While the calls and responses in this procedure are all made over an API, the data transmission will vary based on the web service being used. A user interface is made for humans to use, whereas an API is made for a computer or application to use.

1.2 Company Profile

Tekact Private Limited technology services and product development organization founded by technology experts with a vision to provide Innovations with open-source technologies. Tekact builds innovative products catering to the web, mobile, and cloud with superior quality measures.

The organization is founded and backed by industry leaders having tenures in the past with Oracle, IBM, HSBC & Yahoo!. The experts in the organization contribute to various open-source technologies and provide innovative solutions and products for enterprises and start-

ups. The team of Architects, Developers, Testers, and Operations are experts in their area of work and are so obsessive and passionate about it.

Services:

1. **Software Development:** Software development services are aimed at product management, engineering, operations, and evolving various software types.

   Big Data: In the current industrial scenario businesses are fueled by data. Organizations hardly manage to take over business-critical information about customers, suppliers, products, and operations. The prevailing majority of businesses are looking for Big Data outsourcing to find the technology which could convert their data into valuable insights and implement it.
   Data Aggregation: The process of compiling raw data from a variety of sources into a unified database in preparation for further analysis is referred to as data aggregation. Data can come from a variety of sources – from RDBMS, files, feeds, social media, etc to name a few. These data have to be aggregated based on certain key data elements and are to be made available for analytics and visualization by tools and downstream systems.

   Data Validation: Before importing and processing the data, it is common to practise to perform what is known as "data validation," which is a process that verifies the correctness and quality of the data you have collected. This is an essential step in the process of building a data lake because the data that is accessible must first be made reliable and consistent before accurate analytics can be performed on it.

   Data Governance: The process and management of ensuring the availability, usability, integrity, and security of data that is utilized within an organization is referred to as "Big Data Governance". It includes all the steps from storing the data to securing it from any mishap. We have expertise in technology, processes and policies to store, protect and use the data to achieve business goals.

Data Analytics &Visualization: Data analytics helps individuals and organizations make sense of data. The raw data from various sources is transformed to provide business value. We support descriptive analytics to understand what has happened in a business; predictive analytics to know what could happen, and prescriptive analytics to act towards what should happen. Data visualization tools help to present information in a very comprehensible manner to support decision-making.

Cloud-based Big Data: Big data and cloud computing are two distinct concepts; however, in recent years, these two ideas have become so intertwined that it is difficult to imagine either existing without the other. It is essential to distinguish between the two ideas and investigate the relationships that exist between them. The provision of computing resources and services on demand is what cloud computing refers to. Scalability, Agility, Costs, Accessibility, and Resilience are some of the advantages that the Cloud brings to Big Data solutions. The cloud for Big Data is public, private ideally – hybrid ones usually suffer due to heavy chatting between the cluster nodes. Our experts deliver cloud agnostic solutions, the customers may use services from any cloud vendors.

2. **Quality Assurance:** With the company, be assured that your applications and services are running accurately, functionally and non-functionally

3. **Consulting:** Be a front runner in your business domain, as we help you outrace the technology and customer trends. Build your applications on the best of the platforms employing the best patterns and practices in the industry. Our experts have a great amount of experience in Big Data, Micro Services, Event-Driven, Kappa/Lambda, and Client-Server architecture. We apply the best of the enterprise patterns to achieve maintainable and performant software.

4. **Build Operate Transfer:** We are your dependable ally when it comes to establishing an offshore development center. Reduce the costs associated with opening a development center by sharing your risks with us. Installing an offshore R&D office for software is implied by the building part. For our clients, we appoint a group of

developers, and we look for a workplace. As a client, you maintain complete control over all activities and make regular payments. Operational management is a part of the operating stage. While we handle reservations, payroll, bookkeeping, and human resources, you may continue to concentrate on your product. Additionally, you receive legal assistance and financial counselling. When the customer decides to take over operational management from us, the transfer stage is completed. Since you have been the project's owner from the start, there has been no transfer. Our staff of specialists will depart once clients can manage the R&D office autonomously.

CHAPTER 2

LITERATURE SURVEY

A detailed analysis of the literature on a subject is what is known as a literature review. When doing a literature review, linked research questions are found, and then one looks to find and analyze pertinent literature to address these research issues. The ability to re-analyze the study's findings can lead to the development of fresh insights, which is one benefit of literature reviews. A literature review summarises and explains the entire and up-to-date body of information on a certain subject that may be found in scholarly publications and journal articles. A student may be required to write a literature review as an independent assignment for a class, or they may be required to write one as part of the introduction to or the planning phase of a larger piece of work, the most common of which is a thesis or a research report. Either way, a student may be required to write a literature review in one of two ways. The type of review that has been written will determine the focus and perspective of the review, as well as the type of hypothesis or thesis argument that has been made. This is because the review will be judged based on the kind of argument that has been made. Reading published literature reviews or the first chapters of theses and dissertations in our own subject area can be a helpful way to gain a better understanding of the distinctions that exist between these two types of documents. By doing so, we are able to examine the framework of their arguments and make notes on the various approaches that they take to addressing the problems.

2.1. Purpose of the Literature Review

1. It provides the reader with convenient access to research on a specific subject by selecting articles or studies of high quality that are pertinent, meaningful, important, and valid and then summarising them into one comprehensive report.

2. It gives an ideal starting point for researchers entering a new field by requiring them to summarise, analyse, and compare the original research in that field.

3. It assures that researchers do not repeat previously completed studies.

4. It can suggest topics to focus on or give hints about the direction that future study should take.

5. It highlights the key findings.

6. The literature's discrepancies, gaps, and inconsistencies are noted.

7. It offers an insightful analysis of the methodologies and approaches utilised by previous researchers.

2.2. Related Works

The section mainly describes related study modes in the area of efficient fire detection. And some of them are listed below.

2.2.1 Microservice: In a microservice architecture, a business domain is broken up into a number of autonomous, self-contained contexts that are regularly constrained. These contexts are then implemented by services that are independently deployable and have loose connections to one another. Netflix, which began transitioning away from its monolithic architecture in 2009, when the term "microservice" did not even exist, is considered to be one of the pioneers of the microservices industry. Applications built using microservices grow horizontally well, both from a technological perspective and in terms of the organizational structure of development teams, which may be maintained more compact and nimbler. There have already been attempts to smoothly include such possibilities into adaptive business process management. Additionally, dividing huge apps into separate microservices gives agile teams the next level of independence, which helps expand agile methodologies. Each team may be working on a unique microservice, and each team develops user stories that are specific to the microservices that they are working on. [1].

In event-driven architecture, when a service performs a task that other services may be interested in, it generates an event, which is a record of the activity. These events are consumed by other services so that they may complete any duties necessitated by the occurrence. In contrast to REST, services that generate requests do not need to know the specifics of the services that consume them. Here's a simple example: When a customer places an order on an ecommerce website, a single "order placed" event is generated and processed by several microservices. It is possible to publish events in a number of methods. For instance, they may be published to a queue that ensures delivery of the event to the proper consumers, or they can be published to a "pub/sub" model stream that broadcasts the event and provides access to any parties with an interest. In either scenario, the producer publishes the event, and the consumer receives and responds appropriately to that event. In certain instances, these two parties may also be referred to as the publisher and the subscriber. This project provides a microservice-based, multi-tenant solution for publishing target-based events. And it will give APIs for users to construct applications that are compatible with several domains.[8]

2.2.2 Spring Boot: The Java-based Spring Boot platform for developing web and enterprise applications, and the degree to which it gives service-oriented architecture flexibility (SOA). One of the issues faced by all Spring-based applications is the degree of complexity given by the application's configurations. Spring Boot simplifies the process of designing and deploying production-ready, standalone Spring apps with minimal configuration needs by leveraging Spring. Spring Boot facilitates the development and launch of standalone, production-quality Spring apps by requiring minimal Spring configuration. Spring Boot simplifies dependency management compared to CRUD web applications by employing a comprehensive yet flexible framework and the related libraries as a single dependency.

This makes it possible for Spring Boot to provide a benefit over CRUD web applications. This gives you access to all of the Spring-related technologies required to initiate projects. This framework offers a variety of additional features, such as embedded servers, security, metrics, health checks, and externalised settings, which are often

included in a variety of applications. Prior to being published to a web server, web programmes are often compressed into.war files. However, Spring Boot applications can be compressed into either.war or.jar files, removing the requirement to install and/or configure the programme on the application server [2].

2.2.3 PostgreSQL: PostgreSQL is an efficient open-source database architecture for demonstrations and social movements. It has established a solid reputation for dependability, information integrity, and correctness as a consequence of more than 15 years of continuous development and an effective design. PostgreSQL, often known as Postgres-SQL, is an open-source social database administration system created by a group of volunteers (DBMS). The source code for PostgreSQL can be accessible at no cost to the user, and the database is not under the ownership of any corporate or other private entity. Michael Stonebraker, a professor of software engineering at the University of California, Berkeley, created PostgreSQL.

PostgreSQL was once known by its prior name, Postgres. Postgres was launched by Stonebraker in 1986 as a successor to Ingres, which is now owned and run by Computer Associates. PostgreSQL continues to function normally on all major operating systems, including Windows, Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, and Tru64), and others. It also has programming interfaces for C/C++, Java, Perl, Python, Ruby, and Tcl, in addition to Open Database Connectivity (ODBC). Lists are amazing organisational tools that expedite the retrieval of information considerably. In the great majority of instances, adding a record to a segment will allow you to query the relevant information more rapidly. The trade-off, however, is that the rate at which you embed information will be slower for each record you own. Typically, when you to embed information using a list, you must write the information in two different locations and retain the file's sorting while embedding information. [3].

2.2.4 REST: The proposed solution in this study tried to construct a RESTful API web service. This online service will undoubtedly make it easier to construct software

applications that use programming languages or platforms other than those supported by the system. The Takeaway application will benefit from this research by receiving a web service architecture that employs RESTful API. Numerous parameters, including as filtering, sorting, selection, and pagination, are used to optimise the URI's efficiency. The Takeaway programme has a website that serves as its backend, and an Android app that serves as its frontend. The results of the tests, which were carried out using the Postman application and based on the function method, show that the REST API Server that was installed on the server is functioning properly. Throughout the test, the Apache JMeter application exhibits the best possible response time. Meanwhile, an examination of how long it takes SOAP and REST systems to process responses and queries finds that REST is substantially faster. [4].

2.2.5 Swagger: The OpenAPI or Swagger definition has emerged as the standard for documenting web APIs (REST, RESTful, etc.). As a result, this specification has been and continues to be the subject of extensive research, both in the past and in the present. With this specification, several processes, tasks, and problems related with software development can be enhanced, simplified, solved, and/or automated. The goal of this research is to synthesis the OpenAPI/Swagger specification's uses and applications. It is recognised the development activities in which it has been useful, the software artefacts (results) generated from it, and the domains covered, among other things. We examined 47 articles published between 2011 and 2020 in the literature review. The criteria for the search were as follows: (a) indicating the development activities where the specification was useful; (b) including the software artefacts (results) that it produces and their respective validation; and (c) a reference to the domains that were studied.

According to the findings the OpenAPI/Swagger specification has aided a range of software development tasks, the most notable of which is the improvement of documentation (43 percent). Similarly, 77 percent of the papers address the requirements and/or challenges of web API users. In 68 percent of the papers, task automation based on various tools is the most prevalent result. Furthermore, 40% of the publications consulted cover a specific domain (IoT, Cloud, etc.) [5].

2.2.6 Maven: Maven is an effective piece of software for managing projects, and it is based on POM (project object model). It is utilised in the construction of the project, as well as its documentation and dependency management. Maven makes the day-to-day work of Java developers easier and, more generally, helps with better understanding of any Java-based project. Maven repositories are made up of directories that contain bundled JAR files along with some metadata. POM files, which are connected to the projects that each packaged JAR file is a part of, are where the metadata is stored. This metadata provides a list of the external dependencies that are required for each packaged JAR file. This metadata enables Maven to download dependencies of your dependencies recursively until all dependencies are downloaded and put into your local machine. The construction of a project using Maven takes place in a series of six steps, each of which is dependent on the one that came before it. The following is a list of the phases that are included in the standard Maven lifecycle: - Validate ensures that all of the configuration settings that are essential to the construction of the project have been specified in the appropriate manner. The compilation step transforms the project's source code into deliverables that have not been processed (e.g., .class files). Compiling and running test suites allows testers to search for errors. The target deliverable format is bundled together with the raw deliverables to form packaging (e.g., .jar, .war, .ear). The deliverables are placed in their intended location(s) on the local system when the install is completed. The local installation is transmitted to production environments when you use the deploy command [6].

2.2.7 Postman: In Postman, a test is simply just some JavaScript code that is executed after a request is sent and a response from the server is obtained. The application of POSTMAN is a straightforward character. It provides a set of API calls, and in order to test the application's APIs, one must follow the instructions supplied by that set of API calls. Using the JavaScript programming language, you can design and run tests for each request in Postman. 1. Install and run the Postman programme. There is a native application version available for the macOS, Windows, and Linux operating systems. It is also possible to get a Chrome app for it. If you are using Chrome to access it, you must first click on the square labelled Apps. The most commonly used methods are "GET" for

retrieving data, "POST" for updating data in an existing file, "PUT" for replacing an existing file, and "DELETE" for deleting an existing file (Delete data from the server). Request to Send: a. In the header's toolbar, click the "New" button. a. Enter the Request URL (also known as the Rest Endpoint URL) in the following format: postman-echo.com/get. Users can save requests for later use by clicking the Save button, which is positioned next to the Send button. The Build management tool will be integrated with Newman. It is a Postman command-line companion that allows you to link Postman with a build management solution to enable automated testing. Authorization: By doing a check to verify if you are permitted to do so, the authorization procedure determines whether or not you are authorised to access the data that you want from the server. In that case, we shall proceed to the next step in the procedure. It is part of the security testing procedure. The tool's authorizations come in a variety of flavours, and we'll go through each one individually as we go on to the next section of this study [10].

CHAPTER 3

METHODOLOGY

In “Notify Publisher”, the CRUD (create, read, update and delete) and list operations are done using the Rest API and Spring boot framework. While running it goes through the controller along with the path variables and request parameters that are specified in the URL. After that, it invokes the methods that are declared in the service class and that need to be auto-wired in the controller. The interface then calls the service implementation class and executes the code using the parameters that are specified in the Rest API calls. The implementation invokes respective models and entities respectively. That means when a client sends a request using postman:

- It passes through the controller class
- Then it goes through service class and it gets implemented from the implementation class
- After that it takes respective dto
- Then data goes to postgresql database
- While calling an API , a model can be send as body .
- The model reaches the implementation class and run the code written in the class then it add the model to entity which represent the Database.
- The Repo(interface) is extended by either BaseRepository or JPARepository which include some basic methods like findById(), findAll() etc.
- The Entity class extends BaseEntity which includes some common entities.
- Rendering micro-services.

3.1 Architectural Design :Organization is the parent entity; all other entities are the child entities of organization. Organization has tenants which subscribes for Apps and organization builds Apps. Users belongs to Organization and tenants. Users are assigned to roles and user groups. Users can access the Resources which are linked with Permissions. The structure of

the system, which comprises software components, the externally visible properties of those components, and the relationship between them, is referred to as the software architecture of a computing system. This architecture outlines how the program's primary structural components relate to one another. This modular framework of a computer program can be derived from the analysis models and the interaction of the subsystems within the analysis model. The primary objective is to develop a modular program structure and represent a relationship between the modules. Architectural design is the process of defining the subsystems that make up a system and creating the framework for subsystem control and communication. A description of the software architecture is the result of the design process.

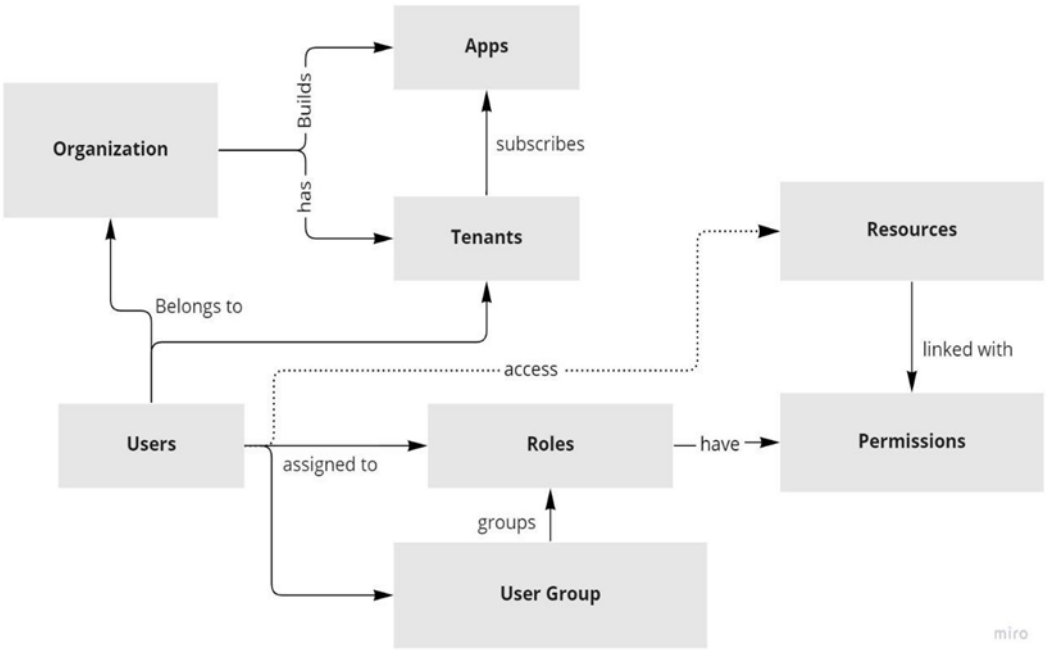


Figure 3.1: Architectural Design

3.2 Module Description

Modules:

- Admin
- Common
- MasterData
- Notify
- Document

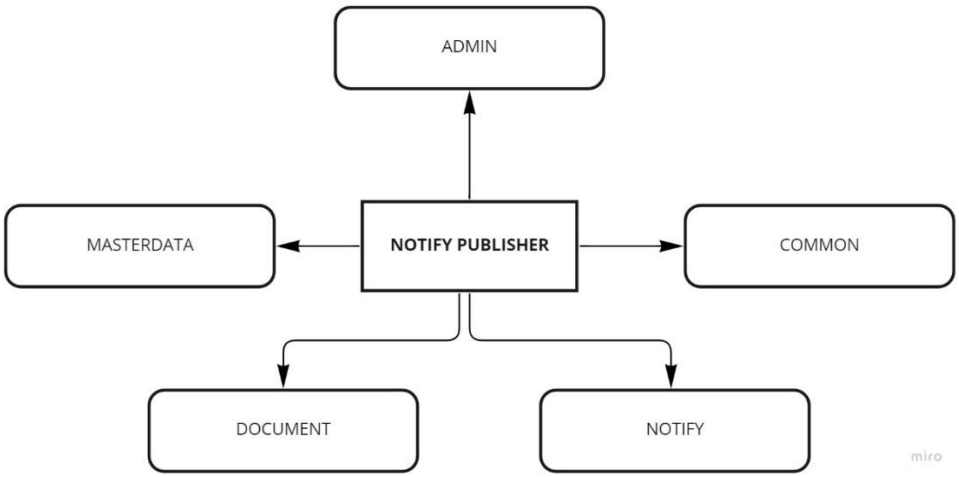


Figure 3.2: Module Design

3.2.1 Admin

Admin module handles all the basic entities like organization, tenant, app, user, user groups, roles. With the help of the Administration module, you can manage all users and groups, install and remove apps, and govern the way Business Process Server runs. The admin functionalities like managing, authorization, authentication of users is all performing under this module. Mainly this Platform service provides the Apis for the organization, tenants and users. The administrative functions like adding, removing, approving, updating users. Also, the adding, removing, approving and updating of organizations, end users, tenants, different user roles and build apps.

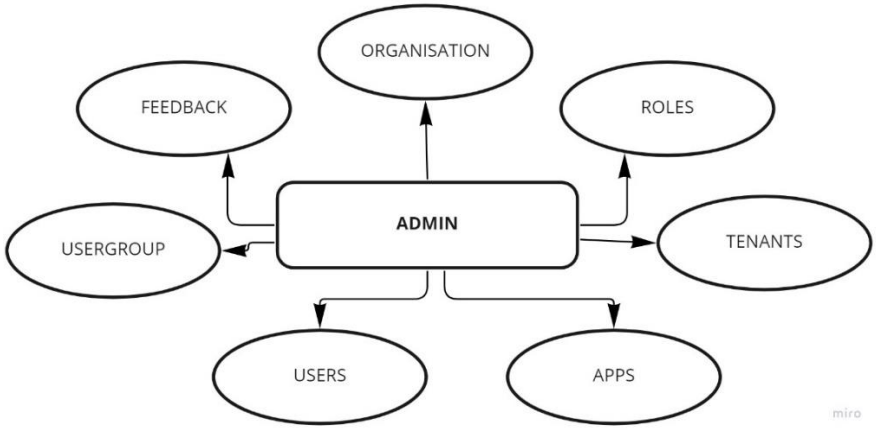


Figure 3.2.1 Admin design

3.2.1.a Organization

Organization is the parent entity of all entities in this project. For example Tekact as an organization it provides various APIs which can be used by tenants to build apps. All other entities like tenant, user, role, app depends on this parent entity. CRUD and List APIs for Organization: As an Admin, one should be able to create, read, update and delete (soft delete) an organization.

API should be available to retrieve all the organizations in the database. Authorization as admin firstly demonstrate creation of the entity then get the entity from the database, update the entity. Soft delete the entity in database And List all the organizations which are active. Deleted organizations should not be listed. The above mentioned apis are created. After that code coverage of at least 80% and no static violations is attained in this section.

3.2.1.b Tenant

A tenant should be able to be created, read, updated, and deleted (soft deleted) by an organisational administrator. API should be available to retrieve all the tenants in the database. Tenant can be a consumer that needs various APIs from the organization. Tenant can be a single user or a complete organization. One organization can have multiple tenants. Tenant subscribes various apps build by the organization.

3.2.1.c Apps

Apps are the products that are created by composing various APIs. The abilities of an organisational admin should include creating, reading, updating, and deleting (soft deleting) apps. API should be available to retrieve all the apps in the database. One also should be able to assign apps to organizations Apps are build by the organization. Such apps are subscribed by the tenants. Apps are controlled and managed by the organization and administrator.

3.2.1.d Users

One organization can have multiple users. They belong to either organization or tenants. Giving the access to the users are controlled by the administrator. Users can be superusers, admin, endusers, tenants etc. One user can be assigned to multiple roles. User can be assigned to usergroup. User is related with feedback entity, userGroupUser entity, userRole entity, tenant entity, organization entity. As an Organizational Admin, the admin should be able to create, read, update and delete (soft delete) users. API should be available to retrieve all the users in the database. Admin also should be able to assign apps to organizations. Validations that is involved in the User is that the App name should be unique and not null.

Acceptance Criteria includes demonstrate creation of the entity, get the entity from the database, update the entity, soft delete the entity in database, list all the tenants which are active, deleted tenants should not be listed, assign apps to Organization, the CRUD operations for the association table of user-usergroup, user-tenant, user-role is also included in the controller, code coverage of atleast 80% and no static violations.

3.2.1.e User Groups

User group is a collection of users with a given set of permissions assigned to the groups and transitively to the users. When the users are associated with a user group, they inherit the permission of the user groups. As an Organizational Admin, one should

have the ability to create, read, update, and delete (soft delete) UserGroups for a Tenant, as well as associate roles with UserGroups. API should be available to retrieve all the user groups for the given tenant in the database. Users also should be able to assign roles to users. Members of the user group can be a user and other groups Eg: Employees, Developers etc

3.2.1.f Roles

Every user has at least one role. Role management and assignment is controlled by administrator. Users and user groups are assigned roles; as an administrator, one should be allowed to create, read, edit, and delete (soft delete) roles. API should be available to retrieve all the roles in the database. Users also should be able to assign roles to users Each role has particular permissions to access resources. Roles are also used for role-based access control

3.2.2 Common

In our project we are following multimodule approach. Thus, for all modules we have common functionalities. All those functionalities used are available in this module. All other module inherits the classes in the common module. Common attributes in all entities, models are mentioned here. This module use base repo and several features and methods. Also, there is common exception class and its own handler are contained in this module, which is used in another modules called admin, master data, document service etc

3.2.3 Document Service

In this module, the system needs to process the input documents and generate meaningful data. Based on this data the system generates events. This story deals with identifying and storing data required for generating the events. The uploaded files (with org, tenant, app, user details) are used for further processing. Acceptance criteria includes, implementing CRUD APIs to for the doc text, DocTextController.

Then implementing approve, delete, listTextByIsApproved APIs. Implement CRUD APIs for doc text trigger. Also implement approved, delete, listTriggerByIsApproved apis for doc text trigger

3.2.4 Master Data

Master module is one of the child of the parent module admin. Any form of dependency between modules and bundles may be established at our discretion. Master module handles entities like country, city, state, contact type entity, organization type entity, address type entity. This module entities use the base entity from the common module, base repo from the common repo and several features and methods from the common module.

The essential information that serves as the foundation for every transaction is known as master data. Whatever your operation may be—producing, moving stock, selling, buying, taking a physical inventory, etc.—it necessitates the maintenance of certain master data. centrally generated data that is accurate for all applications. It doesn't change with time, but we still need to update it frequently. For example: Purchase orders and contracts can be made using the vendor kind of master data.

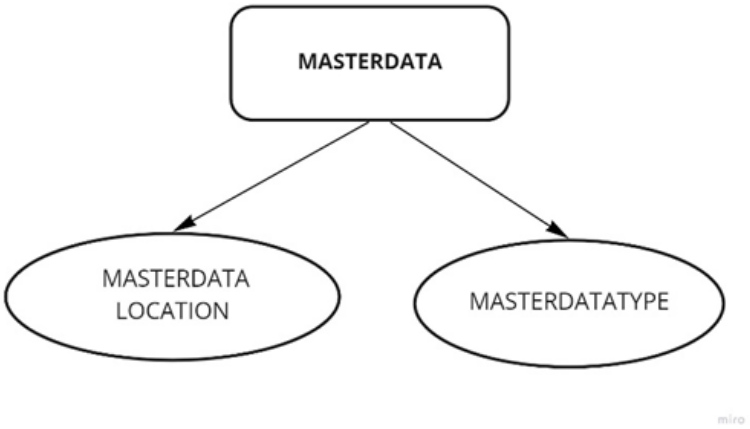


Figure 3.2.4 MasterData Design

3.2.4.a Master Data Location:

For many APIs, we need to specify details of a location. To make sure that no one other than you has logged into your account in a production application, it is crucial for users to examine whose computers, phones, and other devices have recently used their accounts. In order to increase security, the user must be able to identify the location of the devices that have accessed his account in order to be able to erase the connections made on them should it turn out that they were not made by him.

For eg: Address. This entity deals with the creation of Location APIs. Also, the data needed to create the APIs should be developed as Liquibase scripts. In MasterData location we use mainly 3 different entities like city entity, state entity, country entity inorder to achieve the requirements.

Acceptance Criteria includes Liquibase scripts for master data corresponding to Country, State, City, get APIs in LocationController like get all Countries, get states for a given country, get cities in a given state, use cache to store the info at the server side (last priority), the mentioned entities like country, state and city are related to each other such that CRUD operation are performed according with that, test case with code coverage of atleast 80% and no static violations.

3.2.4.b Master Data Type:

In Master data type we need to specify details of a master data types- organization Type, address Type, contact Type, user Type. This story deals with the creation of Master Data APIs. Also, the data needed to create the APIs should be developed as Liquibase scripts. Liquibase is an open-source solution for managing revisions of our database schema scripts. It supports a variety of file formats for defining the DB structure and functions with a wide range of database types. Also, the data needed to create the APIs should be developed as Liquibase scripts.

3.3 RESTful Webservice

REST uses the HTTP Protocol and is based on web standards. Every component is a resource, and each resource is accessed through a common interface utilizing HTTP standard techniques.

3.3.1 Working

It is useful to divide the process into two steps in order to comprehend how RESTful APIs convey data between clients and applications.

3.3.2 Client Request

An application or anybody who uses the API's services is referred to as a client. For instance, if you were integrating with a web platform like Instagram or YouTube, your program would be the client. If you were putting in a URL request, your browser would be the client.

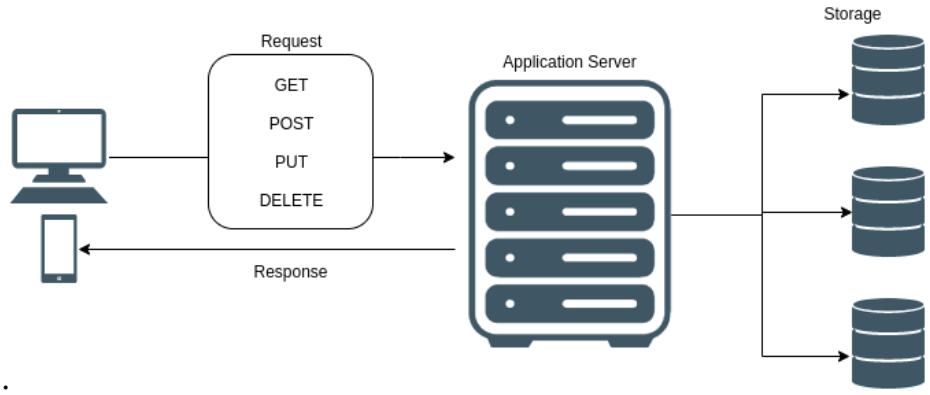


Figure 3.3: REST API Architecture

Examples of HTTP requests include sending an HTTP GET request to Instagram's application programming interface (API) and sending an HTTP POST request to

submit a new photo. On a call centre platform that integrates with an auto-attendant application, the PUT command can be used to update a custom greeting, and the delete command can be used to delete it. This is analogous to the situation described above

3.3.3 Server Response

When an API gives a client specific data, it is referred to as a resource. Anything from hashtags to profiles, comments, web links, tweets, etc. might be included in this. Each resource is stored on a server and has a special name called a resource identifier. When a client uses a RESTful API to submit a request, the server sends the client's system a standardised representation of the resource's state. This means that instead of sending the client the actual resource, the server instead sends a representation of the resource's state, or the status of the resource at a specific timestamp. To make responses easier to understand, they are returned in a simple format. The JSON (JavaScript Object Notation) format is well-liked because it excels at boosting web accessibility and can be read by both people and machines. Additionally, it works with a lot of additional programming languages. Alternative API data formats include plain text, XML, YAML, CSV, and HTML.

3.4 System Specifications

The application development architecture recognized for this project is specified in this section on the basis of requirements.

3.4.1 Software Specification

- Programming Language: JAVA
- IDE: IntelliJ
- Framework: REST API
- Web Browser: Any web browser
- Database: PostgreSQL

3.4.2 Hardware Specification

A computer with internet connectivity and with minimum 4GB Ram

3.4.3 Server Specification

Using aws t2.xlarge instances for each module (with redundancy) vCPU 4, 16GB RAM. Storage will be EBS based 5 GB per instance. Database will be hosted on AWS using Postgres RDS. Amazon.com offers a platform for cloud computing that is referred to as Amazon web services (AWS). This platform is comprised of a number of remote computing services that are also referred to as web services. These services are hosted in eleven distinct geographical locations spread across the globe. The Amazon Elastic Compute Cloud is arguably the most significant and well-known of these services, followed by the Amazon Simple Storage Service. Amazon provides large computing capacity to its customers through the sale of these products as a service, which allows the company to do so more quickly and at a lower cost than if the customers built their own actual physical server farm. In order to carry out the software as a service (SAAS) methodology, this undertaking makes use of Amazon Web Services (AWS) as a server cloud processing platform. A wide variety of services, including every Apache service extension that we employ, are all included in AWS's offering.

3.4.4 Software Description

- **JAVA**

Java is an object-oriented programming language that can create software for multiple platforms. Java was first released in 1995. The code that is compiled when a Java application is written is referred to as "bytecode," and it is compatible with the majority of operating systems (OS), including Windows, Linux, and Mac OS. The computer programming languages C and C++ have had a significant impact on Java's syntax. The programming language Java was developed in the middle of the 1990s by Mike

Sheridan, Patrick Naughton, and James A. Gosling, who had previously worked as a computer scientist for Sun Microsystems. Because Java can create applets, which are browser-based programmes, users on the internet have an easier time interacting with objects and using graphical user interfaces (GUIs). Web pages were largely static and did not support user interaction before the advent of Java applets. Java applets have seen some of their popularity wane in recent years as a direct result of the release of competing technologies such as Adobe Flash and Microsoft Silverlight.

Java applets can only be executed in web browsers when the Java Virtual Machine, abbreviated as JVM, is present. The JVM is responsible for converting Java bytecode into native processor instructions and enabling indirect OS or platform application execution. The Java Virtual Machine (JVM) is responsible for providing the vast majority of the necessary components for the execution of bytecode, which is typically more compact than executable programmes developed with other programming languages. Execution of bytecode is impossible on a system that lacks the necessary JVM. In order to create Java programmes, you will need a Java software development kit (SDK), which typically includes a compiler, an interpreter, a documentation generator, and a number of other tools that are utilised in the process of developing an entire application.

• **INTELLIJ IDEA**

When working with Java and other JVM languages like Kotlin, Scala, and Groovy on a variety of applications, IntelliJ IDEA provides a smart, context-aware IDE. Additionally, IntelliJ IDEA Ultimate's robust integrated tools, support for JavaScript and associated technologies, and improved support for well-known frameworks like Spring, Spring Boot, Jakarta EE, Micronaut, Quarkus, and Helidon can assist you in creating full-stack web applications. Additionally, you may expand IntelliJ IDEA with free JetBrains plugins that enable you to work with additional programming languages including Go, Python, SQL, Ruby, and PHP.

• SPRING BOOT

Spring Framework (Spring Framework) is a popular, enterprise-level, open-source framework for developing independent, high-quality Java Virtual Machine applications (JVM). Java Spring Boot, often known as Spring Boot, is a solution that accelerates and simplifies the development of web applications and microservices utilising the Spring Framework thanks to its three core features. These features are as follows:

1. Autoconfiguration
2. A configuration strategy that is based on personal opinion
3. The capacity to develop applications that run independently

Dependency injection is a feature of the Spring Framework that allows objects to specify their own dependencies, which are then subsequently injected into them by the Spring container. This feature is known as "dependency injection." As a consequence of this, programmers are able to create modular programmes that have components that are only loosely connected to one another. These programmes are ideal for use in distributed network applications and microservices. In addition, the Spring Framework comes with built-in support for many of the common activities that an application must complete. These activities include data binding, type conversion, validation, exception handling, resource and event management, internationalisation, and a great deal more.

FEATURES OF SPRING

Autoconfiguration:-

Applications are initialized with pre-set dependencies through autoconfiguration so you don't have to explicitly set up them. Due to built-in autoconfiguration features, Java Spring Boot configures the underlying Spring Framework and third-party packages according to your settings (and based on best practices, which helps avoid errors). Despite the fact that you can change these defaults after initialization is finished, Java Spring Boot's autoconfiguration feature makes it easier for you to get started creating Spring-based apps quickly and lowers the likelihood of human error.

Opinionated approach:-

Based on the requirements of your project, Spring Boot takes an opinionated approach to add and setting startup dependencies. Instead of having you to make all those decisions and configure everything manually, Spring Boot decides which packages to

install and which default values to use. During the startup process, when you select from a variety of starter dependencies known as Spring Starters that address common use cases, you can specify the requirements of your project. You can use Spring Boot Initialize without writing any code by completing a short web form.

Standalone applications:-

Spring Boot enables programmers to build ready-to-use applications. One can build standalone applications that function on their own without depending on an external web server if you integrate a web server like Tomcat or Netty into your app during the startup phase. This gives you the specific ability to create applications that can function independently.As a result, by just selecting the Run command, you may run your programme on any platform. (To create apps without an embedded Web server, you can choose to disable this capability.)

• **REST API**

An API implementation that complies with the REST architectural restrictions is known as a REST API. It serves as a conduit. Over HTTP, the client and server communicate with one another. The HTTP protocols are used by a REST API to establish communication between the client and the server. REST also gives servers the option to cache responses, which boosts the efficiency of the application. Stateless communication exists between the client and the server. That means that each communication between the client and server is equivalent to a brand-new one. No data or memory from the earlier conversations has been retained. As a result, each time a client communicates with the backend, the client must also send the authentication data. This makes it possible for the backend to determine whether or not the client has permission to view the data. The client interacts with the backend endpoints when a

REST API is implemented. This completely separates the client code from the backend code. Create, Read/Retrieve, Update, and Delete are all parts of the CRUD acronym. These are the persistent storage's four fundamental roles. The CRUD operation is a set of user interface patterns that enable viewing, searching for, and editing data through forms and reports that are computer-based. Data-oriented and employing HTTP action verbs consistently, CRUD.HTTP has a few important verbs.
Standard CRUD Operation

• CREATE Operation: It executes the INSERT statement to create a new record.
• READ Operation: This operation reads table records according to the supplied parameter.
• UPDATE Operation: It executes a table update statement. It depends on the parameter input.
• DELETE Operation: This deletes a row from the table. Additionally, it depends on the input parameter

• **JUNIT + MOCKITO**

Developers utilize JUnit, a Regression Testing Framework, to implement unit testing in Java, speed up programming, and improve code quality. The following can be readily integrated with the JUnit Framework:

- Ant
- Maven
- Eclipse

The following crucial characteristics are provided by the JUnit test framework. Fixtures:- Fixtures are a group of objects in a fixed state that are used as a starting point for tests. A test fixture's main function is to guarantee that tests are executed in a known, consistent environment for reproducible results. It includes the setUp() method, which is invoked before each and every test. method known as tearDown(), which is executed after each and every test method. Test Suites:- A test suite is a collection of individual unit test cases that are executed all at once. Both the

@RunWith annotation and the @Suite annotation are utilised in JUnit in order to execute the suite test. Given below is an example that uses TestJunit1 & TestJunit2 test classes. Test Runners:- The test cases are conducted using a test runner. Here is an example that assumes the test class Test Junit already exists. JUnit Classes: - JUnit classes are crucial since they are utilised while creating and testing JUnits. Some of the important classes are −Assert – Which contains a set of assert methods. TestCase: Consists of a test case that specifies the fixture needed to execute several tests. The class Test Result has methods for gathering the outcomes of running a test case. A mocking framework for Java applications called Mockito is used for unit testing. The creation of testable apps requires the use of mockito. As an open-source testing framework, Mockito was made available under the MIT (Massachusetts Institute of Technology) License. Internally, it creates mock objects for a certain interface using the Java Reflection API. The dummy or proxy objects used in actual implementations are referred to as mock objects. By simulating external dependencies and utilising them in the test code, the Mockito framework aims to streamline the construction of tests.

• **PostgresSQL**

PostgreSQL is a robust, enterprise-class open source relational database that permits querying and supports both SQL (relational) and JSON (non-relational) formats. Its high levels of resilience, integrity, and correctness are due to its very trustworthy database management system. Many web, mobile, GIS, and analytics applications rely on PostgreSQL as their primary data storage or data warehouse. PostgreSQL 12 is the most recent major version. The most common PostgreSQL use cases are mentioned below. LAPP, which stands for Linux, Apache, PostgreSQL, and PHP, is a reliable database system (or Python and Perl). Most dynamic websites and online apps are driven by PostgreSQL, a dependable back-end database. Both established enterprises and start-ups rely on PostgreSQL as their primary database to support their software and merchandise. PostgreSQL supports the following programming languages: Python, Java, C#, C/C+, Ruby, JavaScript (Node.js), Perl, Go, Tcl.

PostgreSQL feature highlights. PostgreSQL has many advanced features that other enterprise-class database management systems offer, such as User-defined types, Table inheritance, Sophisticated locking mechanism, Foreign key referential integrity, Views, rules, and subquery Nested transactions (save points), Multi-version concurrency control (MVCC), Asynchronous replication.

3.5 System Design

A multi-step process that is a representation of data structure, programme structures, interface characteristics, and procedural detail that is synthesised from information requirements has been described as the design. Design serves as a foundation for all software engineering and maintenance step that follows. It is a decision-making activity, frequently of a structural kind. The design creates cohesive, well-planned representations of programmes that focus on the interrelationships of pieces at the upper level and the logical processes at the lower ones.. Depending on the applications and project requirements a good design is one, which allows efficient code to be produced and whose implementation is compact as possible Design elements often contain functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These elements explain the required software features in detail. These design aspects are meant to sufficiently define the software so that skilled programmers may create it with little extra input design. The fundamental design concept specifies the following methods to develop a project: – Abstraction – Modularity – Software Architecture – Structural Partitioning – Data Structure – Software Procedure There are two levels of system design: – Logical design – Physical in logical design, the designer creates a specification of the main system components that achieve the goals. The physical design gives the actual design of the system.

3.5.1 Logical Design

The analysis model clearly demonstrates to the user how the system will operate. This is a system's first technical depiction. Three main goals must be accomplished by the analytical modelling. to create a foundation for software design. To describe what the

user requires To establish a set of specifications that can be tested after the software is developed. Use-case diagrams are used to illustrate how a system is dynamic. This description, however, is too generic to reflect the idea. Because the four other diagrams (activity, sequence, cooperation, and State chart) perform the same role, we will look into a special aim that will set it different from the other four. Use case diagrams are used to collect information about a system's demands, including internal and external aspects. The majority of these requirements are for design. Use case diagrams are used to capture requirements for a system, including those for internal and external parts. These specifications are mostly for the design. Use case diagrams are used to collect information about a system's demands, including internal and external aspects. The majority of these requirements are for the design. As a result, use cases are produced when a system's functionality is determined and actors are defined. After the primary work is completed, use case diagrams are created to depict the outside perspective. As a result, below are some short applications for use case diagrams: used for gathering system requirements utilised to acquire an outside viewpoint on a system Determine the external and internal elements that influence the system. Display the interplay of the criteria as actors.

3.6 Input Design

The process of translating user-generated inputs to a computer-based representation is known as input design. The design of an effective and understandable output should both strengthen the relationship between the user and the system and aid in decision-making. Considering that the management makes use of the design decisions regarding handling input specify how data are accepted for computer processing. The input design component of the overall system design requires careful attention and entails specifying the means by which actions are to be carried out. This part of the design process is an essential component. The most time-consuming and labor-intensive aspect of the system is the gathering of the input data, both in terms of the necessary tools and the number of people engaged. During the input design phase, data are taken in for processing on a computer, and input to the system is carried out through mapping with the assistance of some map support or linkages. The most prevalent source of

errors that occur during data processing is due to inaccurate input data.The design of input is concerned with the following points.

- What data to input?

The initial step of input design is to determine what data to input. In data input ‘Tekact Platform’ uses JSON format and provides the same kinds of advantages as XML does when it comes to exchanging data in a setting that contains a variety of different types of systems, including the following: JSON is self-describing. Applications that are not yet familiar with the data that should be anticipated may, in some circumstances, be able to comprehend the syntax and hierarchical structure of the JSON strings. JSON makes the representation of text very easy. Due to the fact that more complicated document types are not easily shared between platforms and operating systems, it is suitable and secure for doing so. JSON may be easily displayed and edited in basic editors as text. JSON is short. When compared to the size of equivalent XML data, the typical length of a JSON string is approximately two-thirds smaller. JSON is simple to read, simple to grasp, and simple to learn.

The objectives of the input design are:

1. Controlling amount
2. Avoiding delay.
3. Avoiding errors.
4. Avoiding extra steps

In this ‘Tekact Platform’, all the text boxes are validated. If any mandatory field is not filled then it will display the message.

The features of the input design are:

- The input designing is done so as to have the most efficient way for interaction between the user and the system.
- Measures have been taken to minimize user inputs.
- Extra steps are eliminated and the process is made simple**.**

3.7 Output Design

Output refers to the results and information that are generated by the system. Here determines information be present decide the layout and select output medium, arrange a presentation of information in an accepted format and decide how to distribute output to intend recipients Location characteristics and format of column headings and pagination are specified. Output design plays a major role in providing the user with the required format. The major function of the output is to convey information and so its layout and design are careful considerations. The information must be carefully considered to the needs of the user. Standards for printed output suggest giving each Output a name or title, providing a sample of the output layout, and specifying the procedure for providing the accuracy of the output data.

The output devices to consider depends on the compatibility of the devices with the system, response time requirement and printed quality required. In the design output form, attention is given to proper identification and wording, readability and use, composition and layout, order of data items and clarity of instructions. A well-designed form with clarity stated captions should be self-instructing. An organization's form must be centrally controlled for efficient handling. The output of a computer is the most significant and immediate source of information for the user. Output design is a process that involves designing necessary outputs in the form of reports that should give to the users according to the requirements. The design of an effective and understandable output should both strengthen the relationship between the user and the system and aid in decision-making. Because the management uses the reports as a guide for making decisions and drawing conclusions, they need to be carefully designed, and the details contained within the reports need to be straightforward, illustrative, and understandable to the person reading them. Consequently, when it comes to developing output, the following considerations need to be taken into account.

• Choose what information to present to the audience.

• Organize the presentation of the information so that it follows a format that is acceptable.

• Determine how the output will be divided among the intended receipts.
• They are displayed on the monitor for immediate need and for obtaining the hardcopy depending on the nature of the output that is required and the future use that will be required of it.

Efficient and intelligent Output design should improve system relations with the user and help in decision making. That is, this makes the system user-friendly to be displayed or printed as per the user's choice. Quality output is One which meets the requirements of the end user and which presents the information in a way that is clear, easy to read and visually attractive. In order to decide on an appropriate method of presentation and a suitable format, a number of issues to be considered like who receives the output, under what circumstances the output is received, etc.

CHAPTER 4

RESULT AND DISCUSSION

Many organizations nowadays are driving the technology innovations and creating reusable services/apis enables faster time to market for new products. This microservices based project demonstrates how multiple services run independently leveraging on the best microservices patterns to enable scale, performance and resilience. The project aims to achieve the following for the EventPub provide APIs to create Apps, provide ways to compose apps using such APIs, easy integration via usage of REST APIs, workflow management and provide a working App.

This is a product based on microservices architecture where each microservice handles problems specific to a particular domain. The users will be creating apps by stitching those various microservices. This product aims to provide APIs for users to compose Apps that can work with various domains. EventPub is a multi tenant system. It follows a software engineering approach that focuses on decomposing an application into single-function modules with well-defined interfaces There have already been attempts to smoothly include such possibilities into adaptive business process management. Additionally, dividing huge apps into separate microservices gives agile teams the next level of independence, which helps expanding agile methodologies. The microservices that each team works on may vary, and the user stories they create may only have an impact on those microservices.

4.1 Testing methods

There are different types of testing methods available. For this project , testing is implemented to uncover requirements, design or coding errors in the program.

4.1.1 Unit Testing with Mockito

The smallest testable components of an application, known as units, are separately and independently examined for appropriate operation as part of the unit testing phase of software development. Unit testing is an important step in the development process, because if done correctly, it can help detect early flaws in code which may be more difficult to find in later testing stages.

Unit testing is a component of test-driven development (TDD), a pragmatic methodology that takes a meticulous approach to building a product by means of continual testing and revision Prior to using other testing techniques like integration testing, this testing method is also the first level of software testing. Unit tests are typically isolated to ensure a unit does not rely on any external code or functions. Though it can be done manually, testing is frequently automated. A well-liked open source framework for mocking objects in software tests is called Mockito. The creation of tests for classes with external dependencies is made significantly easier by using Mockito. A mock object is a fake implementation of a class or an interface. It enables the output of certain method calls to be defined. The system interaction is often recorded, and testing can verify that.

Typically, there are three stages that make up a unit test: the plan, the cases and scripting, and the actual unit test itself. The unit test is written and reviewed in the first stage. The creation of test cases and scripts is the following phase, after which the code is tested. For test-driven development to work, unit tests must first be written that fail. As soon as the test succeeds, they create code and refactor the application. TDD often produces a code base that is explicit and predictable.

4.1.2 Integration Testing

After unit testing, the software testing process moves on to integration testing. Units or individual software components are tested collectively during this testing. The goal of the integration testing level is to identify flaws when integrated components or units interact.

Modules are used in unit testing for testing purposes, and integration testing combines and tests these modules. The Software is created using a variety of software modules that were created by various programmers or coders. Integrity testing is done to ensure that all of the modules are communicating properly.

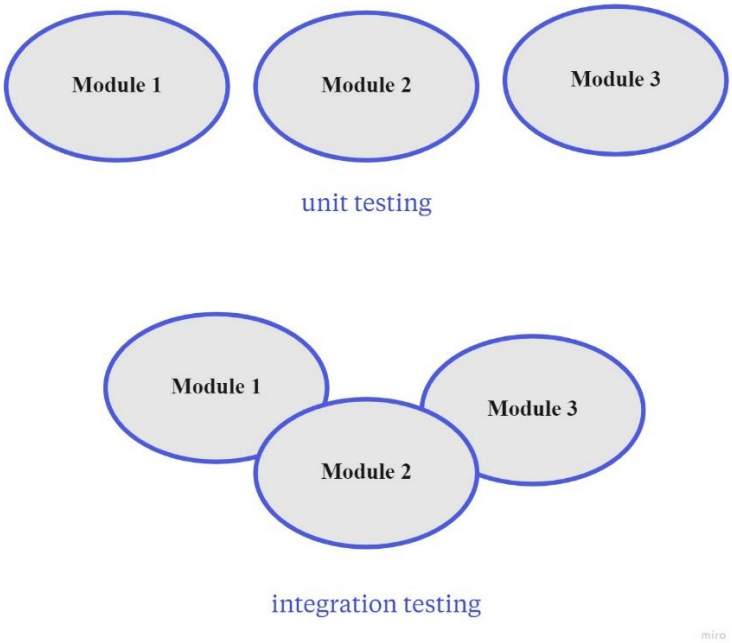


Figure 4.1.2: Testing Modules

4.2 Output Screens and Results

A1.User Controller

As an Admin, one should be able to create, read, update and delete (soft delete) an organization. API should be available to retrieve all the users in the database.

As an Admin one should be able assign a user to specific usergroup

As an Admin one should be able to assign a user to a specific role.

A2. MasterData Controller

API should be available to retrieve all the cities, states and countries As a user one should be able to get the list of the countries.

As a user one should be able to get the list of the states corresponding to the selected country.

As a user one should be able to get the list of the cities corresponding to the selected states.

CHAPTER 5
CONCLUSION

The project aimed to achieve like provide APIs to create Apps, provide ways to compose apps using such APIs, easy integration via usage of REST APIs, workflow management and provide a working App. This product aims to provide APIs for users to compose Apps that can work with various domains. A multi-tenant system. The investigation into the necessity of the system has begun with its first attempt. In order to meet the requirements of the users, a comprehensive study that is also user friendly and straightforward to navigate has been developed. This particular system has been designed in an appealing manner, and the underlying architecture has been simplified to ensure that even a user with a low level of expertise can easily operate the system. This is a very useful system This is a product based on microservices architecture where each microservice handles problems specific to a particular domain. The users will be creating apps by stitching those various microservices. Many firms are pushing technological breakthroughs these days, and establishing reusable services/APIs allows for speedier time to market for new products. This microservices-based project shows how to operate many services separately while employing the finest microservices principles to provide scaling, performance, and resilience.

5.1 Future Enhancement

The system is set up so that adding additional modules can be done without too much trouble. The system's adaptability will enhance as a result of the reconstruction. In order to ensure that the system is as adaptable and simple to use as is humanly possible, the advanced characteristics of this technology were taken into consideration. Easy integration via usage of REST APIs, workflow management and provide a working App.

This helps users in easy way to notify based on certain events. Now a part of backend section is completed for the above project. This application provides various microservices, the users will be creating apps by stitching those various microservices. Also, UI Development (Reactive native for mobile, ReactJS for Web), Cloud hosting (AWS), Containerization (Docker & Kubernetes) will be completed in future.

CHAPTER 6
REFERENCES

[1]. Blinowsk Grzegorz, Ojdowska Anna, and Przybyłek Adam, "Monolithic vs. Microservice Architecture: A Performance and Scalability Assessment", Institute of Computer Science, Warsaw University of Technology IEEE 2022, 00-665 Warsaw, Poland

[2]. Guntupally Kavya," Spring Boot based REST API to Improve Big Scientific Data Report Generation Quality: ARM Data Center Example", 2021 IEEE International Conference on Big Data (Big Data)

[3].Seema Sultana, Dixit Sunanda; "Indexes in PostgreSQL; International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)",2020

[4]. Neumann Andy, Nuno, Bernardino; An Analysis of Public REST Web Service APIs; July 2021 IEEE Transactions on Services Computing

[5].L L Lei, Kai, Ma Yining, and Zhi Tan. "Performance comparison and evaluation of web development technologies in php, python, and node. js." Computational Science and Engineering (CSE), 2021 IEEE 17th International Conference on. IEEE .

[6]. Désarmeaux Casimir, Pecatikov Andrea, and McIntosh."The Dispersion of Build Maintenance Activity across Maven Lifecycle Phases".2021 IEEE/ACM 13th Working Conference on Mining Software Repositories

[7]. M Dheeraj , Sharma Kalpana, "Security Testing of API using Postman and Swagger tools and its use in Internet of Things (IOT)",2020

[8]. I K Aksakalli, T Celik, Ahmet B Can, Bedir Tekinerdogan, "Systematic Approach for Generation of Feasible Deployment Alternatives for Microservices" IEEE Access (Volume: 9)

APPENDIX

SCREENSHOT:

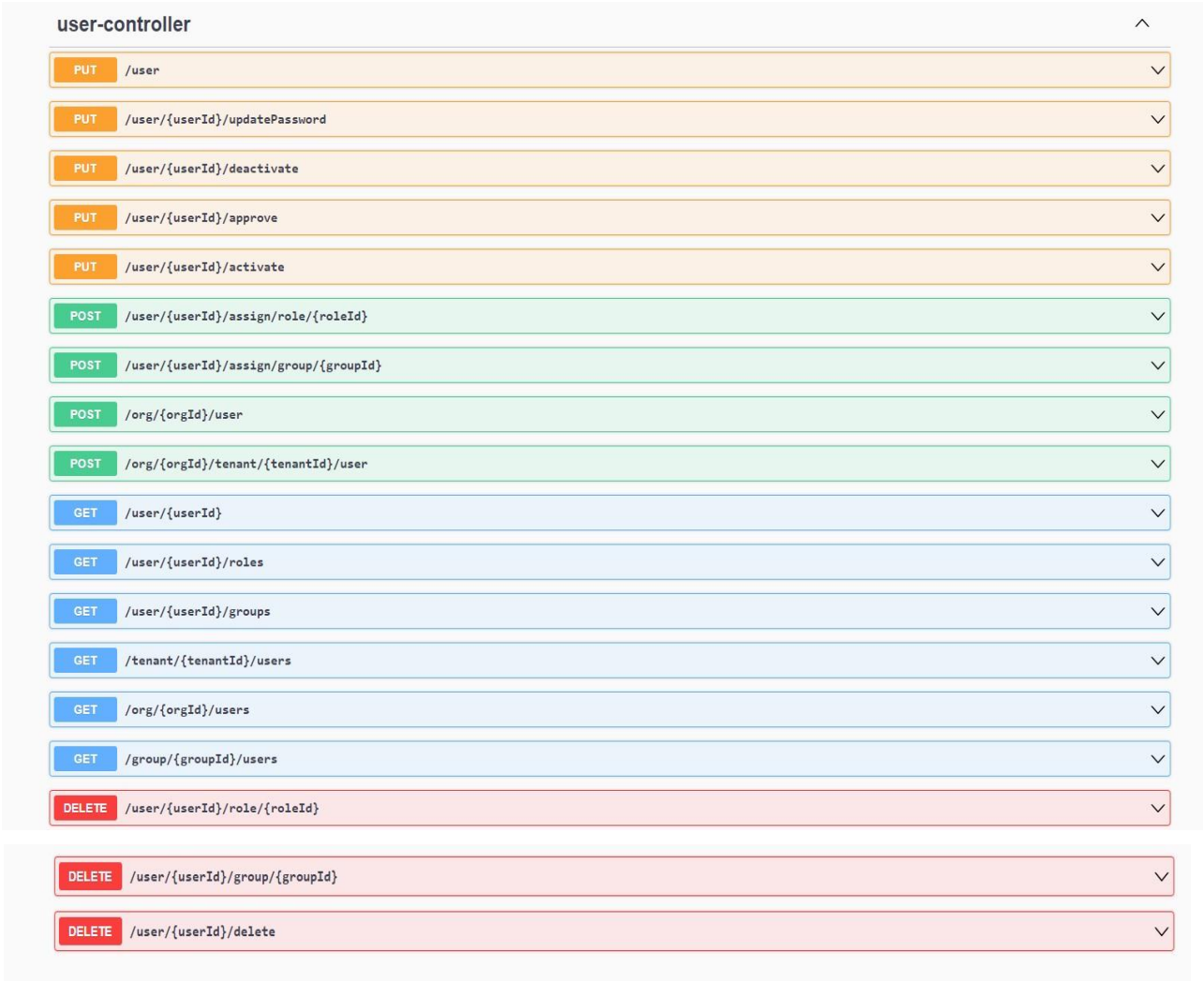
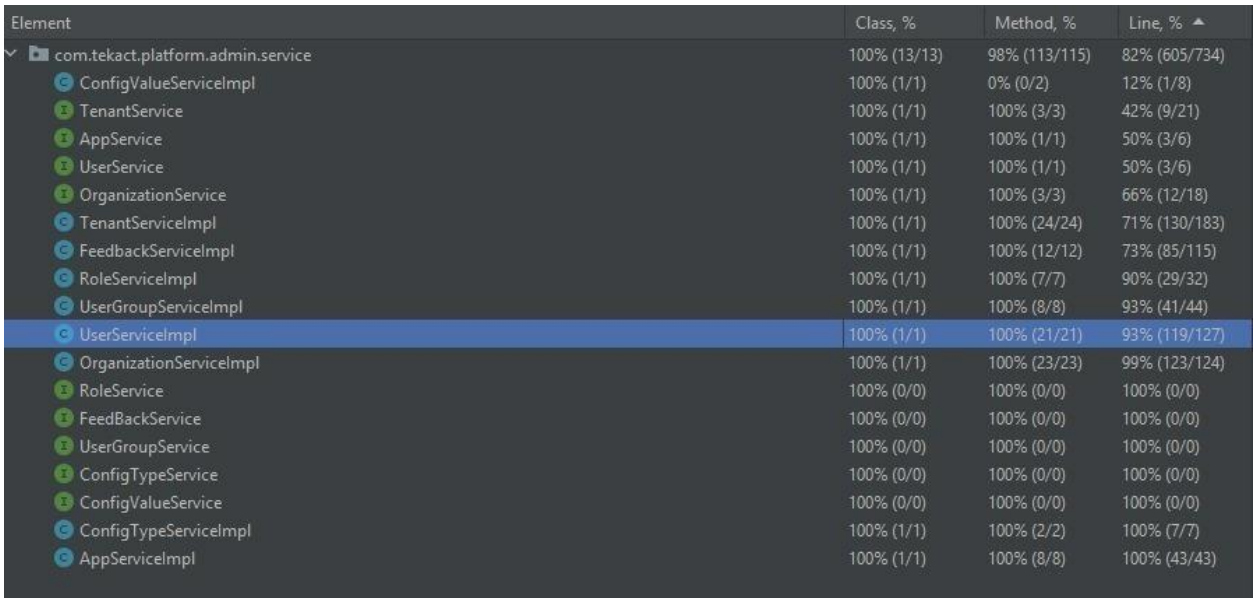


Figure A1. UserController

Figure A2. MasterDataController



Element	Class, %	Method, %	Line, %
com.tekact.platform.admin.service	100% (13/13)	98% (113/115)	82% (605/734)
ConfigValueServiceImpl	100% (1/1)	0% (0/2)	12% (1/8)
TenantService	100% (1/1)	100% (3/3)	42% (9/21)
AppService	100% (1/1)	100% (1/1)	50% (3/6)
UserService	100% (1/1)	100% (1/1)	50% (3/6)
OrganizationService	100% (1/1)	100% (3/3)	66% (12/18)
TenantServiceImpl	100% (1/1)	100% (24/24)	71% (130/183)
FeedbackServiceImpl	100% (1/1)	100% (12/12)	73% (85/115)
RoleServiceImpl	100% (1/1)	100% (7/7)	90% (29/32)
UserGroupServiceImpl	100% (1/1)	100% (8/8)	93% (41/44)
UserServiceImpl	100% (1/1)	100% (21/21)	93% (119/127)
OrganizationServiceImpl	100% (1/1)	100% (23/23)	99% (123/124)
RoleService	100% (0/0)	100% (0/0)	100% (0/0)
FeedBackService	100% (0/0)	100% (0/0)	100% (0/0)
UserGroupService	100% (0/0)	100% (0/0)	100% (0/0)
ConfigTypeService	100% (0/0)	100% (0/0)	100% (0/0)
ConfigValueService	100% (0/0)	100% (0/0)	100% (0/0)
ConfigTypeServiceImpl	100% (1/1)	100% (2/2)	100% (7/7)
AppServiceImpl	100% (1/1)	100% (8/8)	100% (43/43)

Figure A3. Test coverage result of UserController

com.tekact.platform.masterdata.service	100% (3/3)	100% (3/3)	100% (17/17)
CityService	100% (0/0)	100% (0/0)	100% (0/0)
CityServiceImpl	100% (1/1)	100% (1/1)	100% (6/6)
CountryService	100% (0/0)	100% (0/0)	100% (0/0)
CountryServiceImpl	100% (1/1)	100% (1/1)	100% (5/5)
StateService	100% (0/0)	100% (0/0)	100% (0/0)
StateServiceImpl	100% (1/1)	100% (1/1)	100% (6/6)

Figure A4. Test coverage result of MasterDataController