EVENTPUB

A PROJECT REPORT

Submitted by

AMJIYAD AHSAN K S

REG NO: TKM20MCA-2006

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 **EventPub**, 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 AMJIYAD AHSAN K S

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 "**EventPub**" submitted by **AMJIYAD AHSAN K S (TKM20MCA2006),** 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 him 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 Mr. Amjiyad Ahsan 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 he was punctual, hardworking, and inquisitive.

We wish him 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.

AMJIYAD AHSAN K S

ABSTRACT

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 action. These events are consumed by other services so that they can 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 variety of methods. For instance, they can 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 publishes the event and provides access to all parties with an interest. In either scenario, the producer publishes the event, and the consumer receives and responds appropriately to that event. In some instances, these two parties may also be referred to as the publisher and the subscriber. This project provides a microservice-based, multi-tenant product for publishing target-based events. And it will enable APIs for users to create Apps that are compatible with multiple domains.

Contents

List of Figures

Chapter 1
INTRODUCTION

EventPub 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. EventPub is a multi-tenant system. It adheres to a software engineering methodology that emphasises the separation of an application into single-function modules with well-defined interfaces.

These days, many businesses are leading technological advancements, and developing reusable services and APIs speeds up the time it takes for new goods to reach the market. A single application can be developed using the microservice architectural style as a collection of small services, each of which runs in its own process and communicates using simple tools, frequently an HTTP resource API. These services were created with business capabilities in mind and can be independently deployed using deployment equipment that is completely automated. These services, which could be programmed in different programming languages and utilise various data storage methods, are minimally managed centrally.

This microservices-based project illustrates how several services can operate independently by employing the most effective microservices patterns for scalability, performance, and robustness.

1.1 Objective

The project aims to achieve the following for the EventPub product:

1. **Provide APIs to create Apps**

   Application Programming Interface (API) is a software interface that allows two apps to interact with one other without any user intervention. API is a set of software methods and functionalities. API, in its simplest form, refers to executable or accessible software code. API is described as a code that enables two software applications to communicate and share data.

2. **Provide ways to compose apps using such APIs**

   First and foremost, locate an API that your business might utilise. The API key is then used to identify yourself as a legitimate client, to determine access rights, and to log your activities with the API. The simplest way to send a request to an endpoint is to utilise an HTTP client to structure and deliver the request.

3. **Easy integration via usage of REST APIs**

   REST API integration refers to the use of HTTPS requests to access and communicate data with third-party apps. Several cloud services, mobile applications, and IoT devices are appearing on the digital horizon, and organisations are identifying practical applications for these streaming sources.

4. **Workflow management**

   Workflow management is the identification, organisation, and coordination of tasks that generate a given result. Workflow management is the process of simplifying, upgrading, and automating operations wherever feasible to increase output, eliminate duplication, and decrease mistakes.

5. **Provide a working App**

   This project follows the Scrum model to deliver functional applications. Scrum is an Agile project management method that involves a small team led by a Scrum expert whose main job is to remove all obstacles to getting the job done. Work is done in short cycles called sprints, and the group meets daily to discuss ongoing tasks and obstacles that need to be cleared.

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 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 contributes to various open source technologies provide innovative solutions and products for enterprises and startups. The team of Architects, Developers, Testers and Operations are experts in their area of work and are so obsessive and passionate about it. The organization mainly focuses on software development, Quality Assurance, Consulting, Build operate transfer.

Services:

1. **Software development**
    - **Big Data**

      We offer Big Data development that allows organizations to take advantage of many opportunities, such as enhanced information management, improved pricing, better operational efficiency, increased sales and loyalty, greater responsiveness, data driven planning and many more.

- **Data Aggregation**

  Data aggregation is the process through which unprocessed data from various sources is collected and made accessible for additional analytics.
- **Data Validation**

  Data validation is the process of verifying the precision and quality of your data, which is often performed prior to importing and processing.
- **Data Governance**

  Governance of Big Data is the process and administration of the usability, availability, integrity, and security of business data.

- **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.

2. Quality Assurance

- **Functional Test**

  Meet your customer expectations with accurate software. We ensure that the business feature works as per the software requirements
- **Performance Test**

  With our experts ensure that the ability, speed, scalability, and responsiveness of an application holds up, even as your customer size grow
- **Pen Test**

  Evaluate the security of the system - including applications, network & infrastructure, so that you dont fall victim to the vulnerabilities.
- **Automation**

  Automate your quality checks with our Automation experts. This will help you with regressions and assure that the application/services are not broken with rapid changes.
- **API Test**

  Ensure that your APIs are running with accurate functionality, reliability, performance, and security
- **Big Data Test**

  Accurate data helps businesses to better decision making and market advantages. Our experts help the verify and validate the date before its used for business ends

3. Consulting

- **Architecture & Design**

  Build your applications on the best of the platforms employing the best patterns and practices in the industry. We apply the best of the enterprise patterns to achieve maintainable and performant software.
- **Best Practices**

We help your team adopt best of the technology and process practices in the software industry

- **Cloud Computing**

  We help you to identify and gain the best possible business values from cloud computing. We help you with identification of the right cloud vendor, migration, design and implementation of applications in the cloud

4. Build Operate Transfer

- Our experts will help you to build your organization with top notch talent, operate the teams with quality timely deliveries with well equipped process, and transfer the teams to the needy.

Chapter 2
LITERATURE SURVEY

Literary review is the in-depth study and interpretation of literature relevant to a particular topic. Using literature reviews, relevant research questions are identified, and then one seeks to answer those research questions by searching and analyzing relevant literature. A certain importance of literature reviews is that new ideas can be developed by re-analyzing the results of the research. A literature review is both a summary and an explanation of the complete and current state of knowledge on a particular topic as found in academic books and journal articles. There are two types of literary criticism one can write in college: one in which students are asked to write as a standalone exercise in a course, and one that is written as part of an introductory section. introduction or preparation for a longer course. work, usually a thesis or research report. The focus and perspective of the review and the type of hypothesis or thesis will be determined by the type of review that was written. One way to understand the difference between the two is to read published literary reviews or first chapters of theses and dissertations in our field, thereby analyzing their argument structure and Notice how they approach the problem.

2.1. PURPOSE OF THE LITERATURE REVIEW

1. It provides the readers with quick access to information on a specific subject by choosing excellent articles or studies that are pertinent, significant, vital, and valid and compiling them into a single comprehensive report.

2. It provides a good starting point by forcing researchers to start research in a new field to summarize, evaluate, and compare the original work in that particular field.

3. It makes sure that previous work is not duplicated by researchers.

4. It can indicate potential directions for future research or suggest topics to concentrate on.

5. It emphasises the important findings.

6. It clearly demonstrates gaps, discrepancies, and inconsistencies in the literature.

7. It provides an insightful analysis of the procedures and tactics utilised by previous scholars.

2.2. RELATED WORKS

The section mainly describes related technologies used in making the application. All of them are listed below.

2.2.1 MICROSERVICE

A microservice architecture breaks a business domain into small, consistent contexts that are run by services that are independent, self-contained, loosely connected, and can be deployed on their own. Netflix was one of the first companies to use microservices. It started moving away from its monolithic architecture in 2009, before the word "microservice" even existed. Microservice-based applications scale well horizontally, both from a technical point of view and in terms of how the organisation sets up developer teams, which can be kept small and flexible. Adaptive business process management already uses efforts to integrate these kinds of options in a seamless way. Also, breaking up large applications into separate microservices gives agile teams the next level of independence, which helps agile methods scale. Each team may work on different microservices and write user stories that only affect their microservices. [1].

2.2.2 SPRING BOOT

The well-known Java-based Spring Boot framework for creating online and corporate applications, and how it provides the essential flexibility for service-oriented design (SOA). Spring-based application configurations might be tough to comprehend. Spring Boot simplifies the development and deployment of production-ready, standalone Spring applications with minimal setup. Spring Boot streamlines dependency management by using a comprehensive but adaptable framework and the corresponding libraries as a standalone dependency. This gives you access to all the Spring-related technologies necessary to launch projects, which is more than is needed for CRUD web apps. This framework offers additional features that many applications demand, like an embedded server, security, metrics, health checks, and configuration administration. Web applications are often compressed as a war file and sent to

a web server. Spring Boot In contrast, programmes may be packaged as either a war file or a jar file, enabling them to be performed without any installation or configuration on the application server.. [2].

2.2.3 POSTRESQL

PostgreSQL is a robust open-source database foundation for social protests. It has developed a solid reputation for dependability, precision, and informational accuracy during more than 15 years of dynamic design and development. PostgreSQL (pronounced post-gre-SQL) is a social database management system (DBMS) developed by a community of volunteers. PostgreSQL is not under the authority of any other corporate or private body, and the source code is openly accessible. PostgreSQL, formerly known as Postgres, was created at the University of California by software engineering professor Michael Stonebraker. Stonebraker introduced Postgres in 1986 as the successor to Ingres, which is now owned by Computer Associates. PostgreSQL remains compatible with all major desktop frameworks, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru6), Windows, and Mac OS X. It incorporates programming interfaces for C/C, Java, Perl, Python, Ruby, Tcl, and Open Database Connectivity and provides support for content, graphics, audio, and video (ODBC). Lists are useful for quickly locating information. In general, adding records to sections will expedite the retrieval of information. However, the trade-off is that you will merge information more slowly for each record you have. Essentially, when information is added to a list, the list must compose the information in two locations and continue sorting the file as the information is integrated.. [3].

2.2.4 RESTFUL API

The suggested approach in this study attempted to The implementation of a RESTful API web service would undoubtedly make it easier to construct software applications outside of the system or on multiple platforms or with different programming languages. With the use of a RESTful API, this study will create a web service architecture for a takeout application. A number of factors, including as filtering, sorting, selection, and pagination, are employed to

optimise the URI.

An application's backend is a website, while its frontend is an Android-based device. The REST API Server established on the server has been functioning successfully, according to test results based on the function method using the Postman programme. The Apache JMeter application exhibits an excellent response time while testing the response time. In the meanwhile, REST requires less time to complete requests and answers as compared to SOAP systems. [4].

2.2.5 THYMELEAF

Thymeleaf is essentially a template engine for Java that is able to function in non-web contexts and applications in addition to those that are built on servlets. Although it is best suited for use with HTML5 in online applications, it is also capable of performing well in standalone apps. It is possible to use it as a replacement for JSP since its template files may be accessed straight from inside the browser. The primary characteristics are as described below: - It is possible to utilise this for comprehensive documentation. It is possible to utilise it as a template engine framework if that is desired. It is able to determine the kind of document and convert it appropriately at the appropriate time. Because of a function known as parsed template cache, which provides both high speed and efficiency, the amount of input and output data that is processed is cut down to an absolute minimum. It is able to function well in both online and offline settings simultaneously. The fact that templates in this engine function as prototypes makes it possible for us to implement a natural templating system, which is the primary benefit of this engine in contrast to the others that we have. The templates load in the browser without the need for a server to be active; this method of displaying templates is known as "static display." It functions in much the same way as a template engine framework. [5].

2.2.6 SWAGGER

The standard option for defining web APIs is now the OpenAPI or Swagger specification (REST, RESTful, etc.). This standard has been and continues to be the subject of several research studies. With the help of this specification, a number of processes, jobs, and problems pertaining to software development may be streamlined, handled, and/or automated. This

research aims to compile the OpenAPI/Swagger specification's usage and applications. Among other things, it identifies the development activities where it has been helpful, the software artefacts (results) produced from it, and the areas it has covered.

There were a total of 47 publications in the literature review that we looked at that were published between 2011 and 2020. The search was based on the following criteria: a) a description of the development activities where the specification has been helpful; b) a listing of the software artefacts (results) that it produces and the validation of those results; and c) a mention of the domains covered.

The enhancement of the documentation is the activity that has profited from the OpenAPI/Swagger standard the most, according to the results (43 percent). In a similar vein, 77% of the articles provide answers to the needs and/or issues of web API users. In 68 percent of the papers, work automation using a variety of methods is the most common outcome. In addition, 40% of the literature examined deal with a particular subject (IoT, Cloud, etc.) [6].

2.2.7 MAVEN

Maven is a powerful project management tool based on POM (Project Object Model). used for documentation, dependencies, and the construction of projects. Maven streamlines Java developers' daily tasks and often aids in their comprehension of Java-based projects. The Maven repository is a collection of JAR files that have been compressed and include some information. The POM file linked to the project that each packed JAR file is a part of contains the metadata, including a list of each packed JAR file's external dependencies. With the help of this information, Maven is able to download dependencies repeatedly until all of them have been downloaded and saved locally. Maven builds projects in a succession of six build steps, each of which depends on the one before it. The following list of stages represents the fundamental Maven life cycle 1: Validate makes sure that every configuration required to construct the project is active. The project source code is compiled to provide unprocessed results (such as .class files). The test suite is compiled, executed, and error-checked. Packaging incorporates the raw findings into the desired outcome format (.jar, .war, .ear, etc.). The installation stores the outcomes on your local system in the desired area. Local installations are pushed to production through deployment. [7].

2.2.8 POSTMAN

A test in Postman is essentially JavaScript code that is executed when a request is made and a response is received from the server. POSTMAN is highly intuitive. It provides a sequence of API calls that must be executed in order to test an application's APIs. Postman facilitates the creation and execution of JavaScript-based tests for each request. Install the application postman. Native applications are available for macOS, Windows, and Linux. It's also available as a Chrome app. Click the square within the Apps section if you're using Chrome. The most common HTTP methods are GET (data retrieval), POST (updating data in an existing file), PUT (for replacing a file), and DELETE (Delete data from server). Send Request: a. In the toolbar in the header, click New a. Enter the URL for the Request (URL- Rest endpoint) as depicted in the image below. URL is postman-echo.com/get. By clicking the Save button situated next to the Send button, the user can save a request for later use. Combine Newman with the Build management application. It is the Postman command line companion that allows Postman to be integrated with a build management solution to assist automated testing. Authorization: The authorisation method verifies if you have access to the requested server data. It is a component of security testing.[8].

Chapter 3
METHODOLOGY

In “EventPub”, the CRUD (create, read, update and delete) and list operations is done using the Rest API and Springboot framework. While running it goes through the controller along with the path variables and request parameters that specified in the url. After that it invokes the methods that are declared in service class and that needs to be autowired in the controller. The interface then call the service implementation class and execute the code using the parameters that are specified in the Rest API calls. The implementation invokes respective models and entities respectively. That means when 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

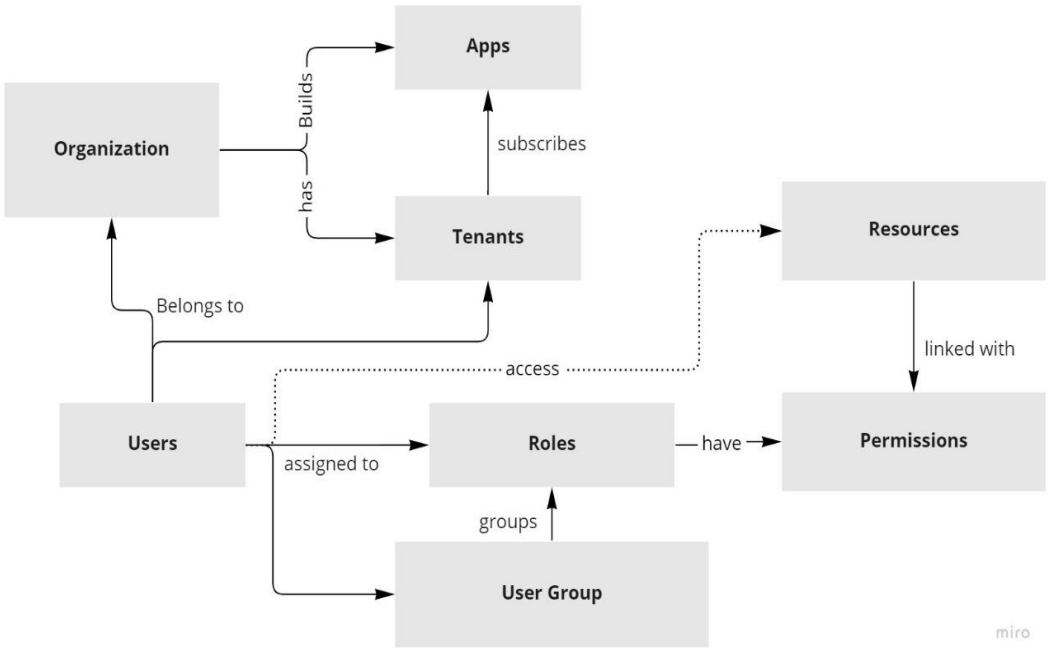


Figure : Architectural Design

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 software architecture of a computer system is the structure of the system, which consists of software components, the externally observable attributes of those components, and their relationships; it describes the link between the program's primary structural pieces. This modular architecture of a computer programme may be constructed from the analysis methods and subsystem interactions inside the analysis model. The basic purpose is to create a modular programme structure and show the modules' connection. Architectural design is the process of determining the subsystems that comprise a system and the framework for subsystem control and communication. This design approach will result in a description of the software architecture.

3.2 Module Description

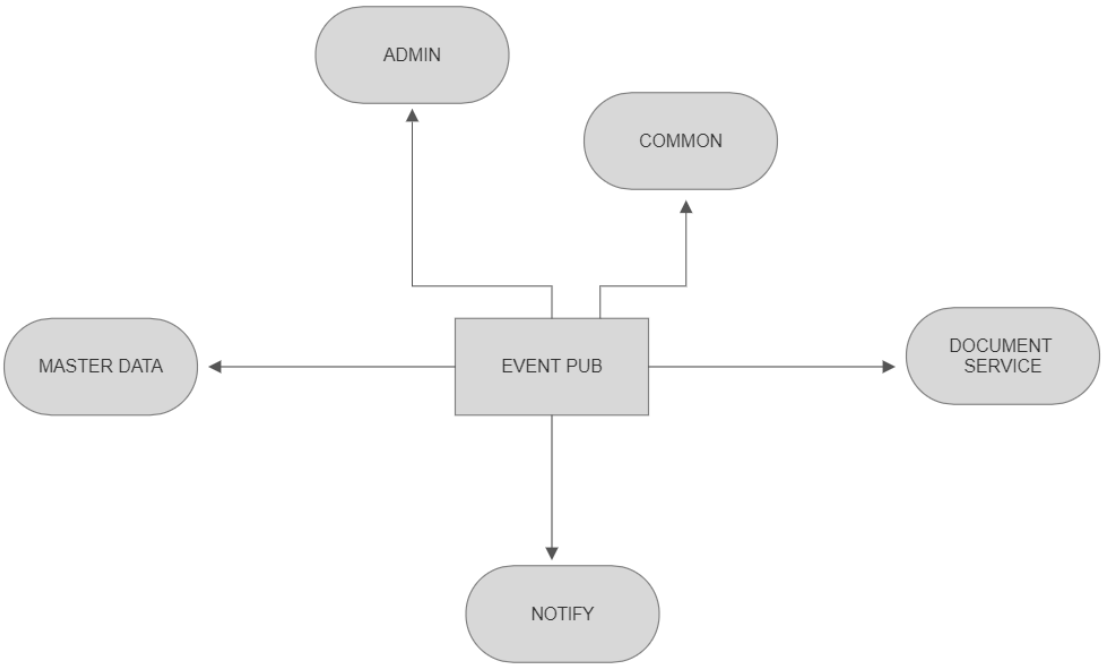


Fig :Module Description

3.2.1 Admin

Admin module handles all the basic entities like organization, tenant, app, user, user groups, roles. The Administration module allows you to control the operation of Business Process Server, install and uninstall applications, and manage all users and groups. 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

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, I should be able to create, view, modify, 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

Tenants are the Consumer that subscribes various Apps built by the organization. Tenants can be a single user or a complete organization. For an organization we can have multiple tenants. Tenant subscribes to various apps build by the organization to achieve their needs. CRUD and List APIs for Tenants: As a Tenant, I should be able to I should be able to create, view, modify, and delete (soft delete) . API should be available to retrieve all the tenants in the database. Tenant have to register by providing the details to database, After the creation of the entity ,get the entity from the database, update the entity. Soft delete the entity in database And List all the tenants which are active. Deleted tenants 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.

CRUD - Address & Contact to Tenant: As an Organizational Admin, I should be able to create, view, modify, and delete (soft delete) address/contacts to an Organization. API should be available to retrieve all the addresses/contacts in the database. authorization as Organizational Admin validation includes App name should be unique and not null. Acceptance Criteria includes Demonstrate creation of the entities (address/contact), Get the entity from the database, Update the entity, List all the addresses/contact of the tenant, Code coverage of atleast 80% and no static violations

3.2.1.c Apps

Apps are the products that are created by composing various APIs. As an Organizational Admin, i should be able to I should be able to create, view, modify, and delete (soft delete) apps. API should be available to retrieve all the apps in the database. I 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, end-users, tenants etc. As an Organizational Admin, I should be able to I should be able to create, view, modify, and delete (soft delete) users. API should be available to retrieve all the users in the database. I also should be able to assign apps to organizations. One user can be assigned to multiple roles. User can be assigned to user group.

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, I should be able to I should be able to create, view, modify, and delete (soft delete) User Groups for a Tenant and associate roles to UserGroups. API should be available to retrieve all the user groups for the given tenant in the database. I 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.

CRUD operation of UserGroup include create, read, update and delete. POST method is used for create usergroup by passing orgid. For updating the values in the UserGroup use PUT method. There two GET method are used here first one is to get all the details of usegroup by passing orgid as pathvariable and the second one is to get the particular details of uergroup by passing usergroup id.Soft delete the entity in database And List all the user group which are active. Deleted user group 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.f Roles

Every user has at least one role. Role management and assignment is controlled by administrator. Roles are assigned to users and user groups, as part of admin It should be able to create, view, modify, and delete (soft delete) roles. API should be available to retrieve all the roles in the database. Also it 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

Document service provides the apis for the documents - to create/upload, update, download, delete, list and to get documents. OCR - Create Metadata from image uploaded.To extract useful data from a given image file, Convert image data to grayscale, smooth, de-skew, and filter, for example. Lines, words, and characters are detected. Create a ranked list of potential character candidates based on a training data set. (here, the setDataPath() method is employed to set the trainer data path). Choose the best identified characters based on confidence from the previous phase and language data. Language data comprises dictionary, grammar rules, etc. It improves the efficacy and productivity of office work. The ability to immediately search through content is incredibly beneficial, particularly in an office environment that must deal with significant volumes of document scanning or document influx. OCR is rapid, maintaining the integrity of the document's content while saving time. Since staff are no longer need to waste time on manual labour, they may work more quickly and efficiently, resulting in an increase in output.

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 MasterData

Master data is the essential information used as the foundation for all transactions. Whether you are producing, transferring inventory, selling, purchasing, or conducting physical inventory, you must maintain certain master data. Data created centrally and applicable to all apps. It remains consistent throughout time, but we must regularly update it. For instance: A vendor is a type of master data used to generate purchase orders and contracts.

Master module is one of the child of the parent module admin. We are free to create any type of dependency between modules and bundles together. 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.

3.1.4.a Master Data Location:

For many APIs, we need to specify details of a location. In a production application, it is important as a user to check which computers, phones, and other devices have recently used your account, to ensure that no one other than you has signed into your account. To improve security, the user must be able to know the geographical position of the equipment which has used his account in order to be able to delete the connections on them in the event that the connections do not come from 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.

3.1.4.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 works across various types of databases, and supports various file formats for defining the DB structure. Also, the data needed to create the APIs should be developed as Liquibase scripts.
Acceptance Criteria includes Liquibase scripts for master data corresponding to organization Type, address Type, contact Type, user Type, get APIs in MasterDataTypeController like get all type for organization Type, address Type, contact Type, user Type ,use cache to store the info at the server side (last priority), the mentioned entities like organization Type, address Type, contact Type, user Type are related to each other such that CRUD operation are performed according with that, test case with code coverage of at least 80% and no static violations.

3.2.5 Notify

This module deals with creating Apis for generating notifications over email. It provides a mechanism for users to get updates related to user onboarding ,user registration and the status. This allows users to get timely updates .Acceptance criteria includes get notify medium API as list, get notify events - CRUD + list by get notify events by entity type, API to send email notifications. The data (key,value) to be replaced in template is to be passed as parameters and List of email ids should provide from the client. Separate template is generated for Feedback, User registration ,Tenant registration ,App and Org registration . Thymeleaf is used for rendering the data and display it in the template.

3.3 RESTful Webservice

REST is an HTTP-based web architecture based on web standards. Each component represents a resource, and standard HTTP methods are used to retrieve the resource via a standardised interface.

3.3.1 Working

To comprehend how the RESTful API transports data between clients and applications, it is helpful to break the process into two distinct phases.

3.3.2 Client Request

A application or person that makes use of an API service is referred to as a client. For instance, if you integrate your software with a web application such as Instagram or YouTube, your programme will act as the client for the combined application. Your web browser will become the client whenever you make a request for a URL. The following are the most frequent HTTP methods:

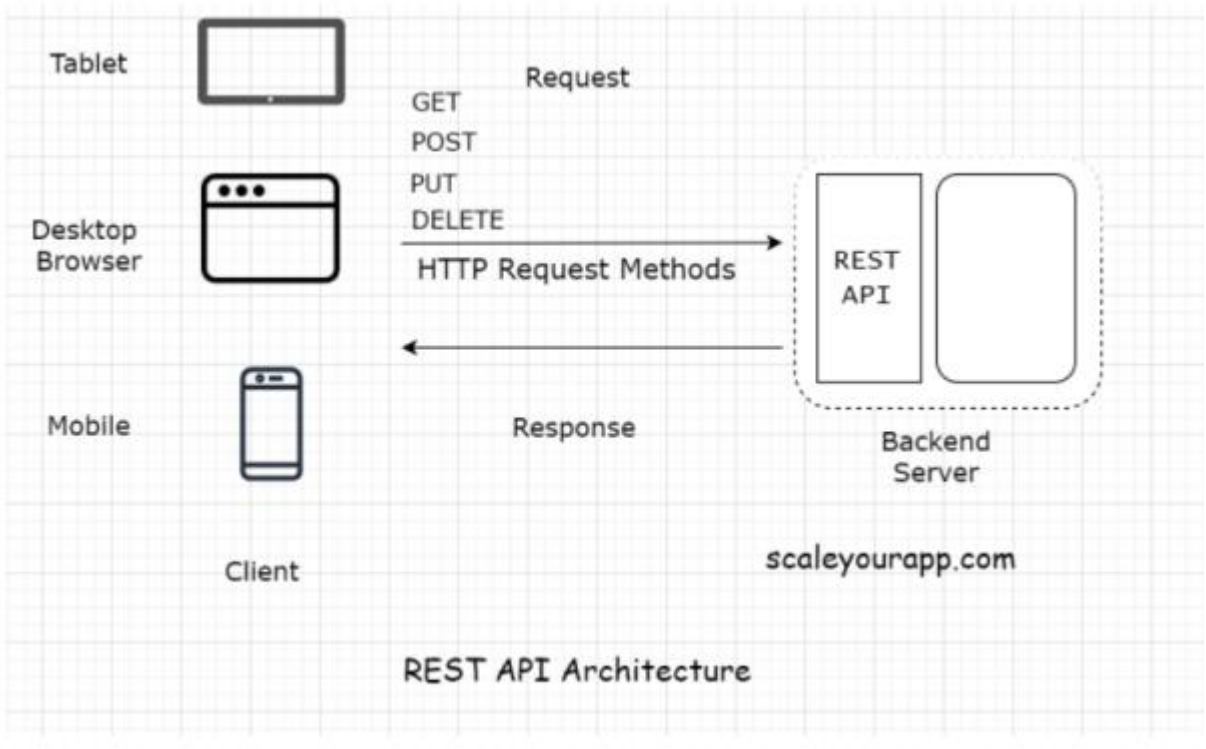
• GET: fetches a resource.

• POST: Make a new resource available to users.

• PUT: Editing or refreshing an already existent resource.

• Take Away: This gets rid of the original source.

This may be an HTTP call to the Instagram API, such as a GET request for a particular video or post or a POST request to upload a new picture. Alternatively, the request may be to delete an existing image. In a similar manner, the contact centre platform that is coupled with the autoresponder programme has the capability to remove or change the custom greeting using the PUT and DELETE commands, respectively.



3.3.3 Server Response

A resource is the particular information provided by an API to a client. This may include hashtags, profiles, comments, site links, tweets, and more. Each resource has a unique identification known as a resource identifier and is kept on a server. When a client uses a RESTful API to make a request, the server sends a standardised representation of the resource's state to the client's system. This indicates that the server does not transmit the client the real resource, but rather a representation of its status at a certain date. Responses are returned in a lightweight manner to facilitate comprehension. JSON (JavaScript Object Notation) is a popular format because it is understandable by both people and computers and excels at boosting online accessibility. Additionally, it is interoperable with several additional programming languages. XML, YAML, CSV, HTML, and plain text are alternate data formats for APIs.

3.4 System Specifications

This section describes the application development architecture identified for this project based on needs.

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) with four vCPU and sixteen gigabytes of RAM. There will be 5 GB of EBS-based storage per instance. AWS will host the Postgres RDS database. Amazon's web services (AWS), a set of remote computing capabilities also known as web services, constitute a cloud-computing platform. These services are offered in eleven geographical zones throughout the globe. Amazon Elastic Compute Cloud and Amazon S3 are perhaps the most prominent and well-known of these services. Amazon sells these goods as a service in order to deliver huge processing power more rapidly and cheaply than a client firm could by constructing a physical server farm. Using the Software as a service (SAAS) methodology, AWS is employed as a Server Cloud Processing platform in this project. AWS offers a multitude of services, including all of the Apache services extensions used by our system.

3.4.4 Software Description

- **JAVA**

  Java is a platform-independent, object-oriented programming language used it to construct programmes on several systems. When a developer creates a Java programme, the resultant code (known as bytecode) is compatible with Windows, Linux, and Mac OS, among others. A significant portion of Java's syntax is borrowed from C and C++. In the mid-1990s, former Sun Microsystems computer scientist James A. Gosling co-created Java with Mike Sheridan and Patrick Naughton.

  Java creates applets (browser-executed programmes) that enable users of the Internet to engage with graphical user interface (GUI) and resources. Before Java applets, the majority of websites were static and non-interactive. Due to the introduction of rival technologies such as Adobe Flash and Microsoft Silverlight, Java applets have lost appeal. Java applets execute on a web browser with the help of the Java Virtual Machine (JVM), which transforms Java bytecode into native processor instructions and allows indirect OS or platform programme execution. JVM supplies most of the components required to run bytecode, which is often smaller than executable programmes developed in other programming languages. If a system lacks the necessary JVM, bytcode cannot be executed. The creation of Java applications needs a Java software development kit (SDK), which consists of a compiler, an interpreter, a documentation generator, and other key tools.

- **INTELLIJ IDEA**

  IntelliJ IDEA is an integrated development environment (IDE) that is smart and aware of its context, and it can be used to work with Java and other languages that run on the JVM, such as Kotlin, Scala, and Groovy, to develop a wide variety of applications. In addition, IntelliJ IDEA Ultimate is capable of assisting you in the development of full-stack web applications. This is made possible by the software's powerful integrated tools, support for JavaScript and other related technologies, and advanced support for widely used frameworks such as Spring, Spring Boot, Jakarta EE, Micronaut, Quarkus, and Helidon. You may also expand IntelliJ IDEA with free plugins built by JetBrains,

which gives you the ability to work with additional programming languages like as Go, Python, SQL, Ruby, and PHP. These plugins can be downloaded from the JetBrains website.

- **SPRING BOOT**

Java Spring Framework (Spring Framework) is a popular, enterprise-level, open-source framework for creating production-ready, stand-alone Java Virtual Machine applications (JVM). Java Spring Boot, is a technology that simplifies and accelerates the development of web applications and microservices using Spring Framework. This is performed primarily in three ways:

1. Autoconfiguration

2. Having an opinion about configuration

3. The ability to make apps that run on their own

Together, these features give you a tool that lets you set up a Spring-based app with as little configuration and setup as possible. Dependency injection is a feature of the Spring Framework that lets objects define their own dependencies, which are then filled in by the Spring container. This makes it possible for developers to make modular applications with loosely connected parts that work well for microservices and distributed network applications.
Spring Framework also has built-in support for tasks that most applications need to do, like data binding, type conversion, validation, handling exceptions, resource and event management, internationalisation, and more.

FEATURES OF SPRING:
Using autoconfiguration, programmes are started with their dependencies already specified, avoiding the need to specify them manually. Java Spring Boot's built-in autoconfiguration automatically configures the base Spring Framework and third-party packages in accordance with the configuration parameters you supply (and based on best practices, which helps avoid errors). Even though you may modify these settings

once startup is complete, the autoconfiguration functionality of Java Spring Boot simplifies the creation of Spring-based apps and reduces the likelihood of human mistake. Spring Boot offers a method to add and configure initial dependencies depending on the needs of your project. Spring Boot uses its best judgement to choose which packages to install and which defaults settings to use. You are not required to manually set each and every one of these options. During the setup process, while choosing a starter, you may outline your project's needs.

Spring Starters are collections of dependencies covering typical usage scenarios. Spring Boot Initializr just requires the completion of a basic online form. You do not require coding knowledge.

Standalone applications:-

Spring Boot makes it easier for developers to make apps that just work. In particular, it lets you make apps that don't need an outside web server to run. You can accomplish this by incorporating a web server such as Tomcat or Netty into your application during the initialization phase. Consequently, you may execute your programme on any platform by pressing the Run command. (You can disable this capability if you wish to develop applications without a Web server)

REST API

An API implementation that complies to the REST architectural requirements is a REST API. It serves as a user interface. HTTP is used for communication between the client and server. A REST API makes use of HTTP protocols to facilitate client-server communication. REST also allows servers to cache the answer, which increases the speed of the application. Client-to-server communication is a stateless process. That is to say, each communication between the client and the server is essentially fresh. No information or recollection is transferred from earlier interactions. Therefore, whenever a client interacts with the backend, the client must also transmit authentication information. This allows the backend to determine whether or not the client is permitted to access the data. In a REST API implementation, the client talks with the backend endpoints. This completely decouples the client code from the backend. Create, Read/Retrieve, Update, and Delete are abbreviations for Create, Read/Retrieve, Update, and Delete, respectively. These are the four fundamental capabilities of persistent storage. The CRUD operation may be characterised as user interface principles that

enable information to be viewed, searched, and modified using computer-based forms and reports. CRUD is data-centric and uses HTTP action verbs consistently. HTTP has many key verbs.

• POST: Produces a brand-new resource

• GET: Reads a resource

• PUT: Changes an existing resource

• DELETE: This operation deletes a resource Standard CRUD Procedure

• INSERT: executes the statement to create a new record.

• READ Operation: This operation retrieves table records based on the given input.

• UPDATE Operation: It executes an update statement on a table. It depends on the input parameter.

• The DELETE operation removes a row from a table. Moreover, it is dependent on the input parameter.

- **JUnit+Mockito**

Developers utilise JUnit, a Regression Testing Framework, to create unit testing in Java, speed up development, and improve code quality. The following can be readily integrated with JUnit Framework:

• Eclipse

• Ant

• Maven

The following crucial characteristics are provided by the JUnit test framework. Fixtures: Fixtures are a collection of objects in a fixed state that are used as a reference point while executing tests. A test fixture's main function is to guarantee that tests are executed in a known, consistent environment for reproducible results. It contains the setUp() function, which is executed before to each test execution. Every test method is followed by the tearDown() function. Test Suites: A test suite combines and executes a number of unit test cases. JUnit uses the annotations @RunWith and @Suite to perform the suite test. An example using the test classes from TestJunit1 and TestJunit2 is shown below. Test Runners: Test cases are executed using test runners. The test class

TestJunit is assumed to be present in the following example. JUnit classes are crucial classes that are needed while creating and testing JUnits. among the crucial subjects are 'Assert' includes a collection of assert methods. TestCase: Consists of a test case that specifies the fixture needed to execute many tests. The class TestResult 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 fake objects for a certain interface using the Java Reflection API. The fake or proxy objects used in real implementations are referred to as mock objects. By simulating external dependencies and using them in the test code, the Mockito framework aims to streamline the construction of tests.

- **PostgresSQL**

Relational database PostgreSQL implements both SQL and JSON querying. Its high levels of endurance, accuracy, and reliability are the result of a very reliable database administration system. Numerous online, mobile, GIS, and analytics applications use PostgreSQL as their main data store or data warehouse. PostgreSQL 12 is the latest significant release. These are typical PostgreSQL applications. A powerful LAPP database. LAPP is an acronym for Linux, Apache, PostgreSQL, and PHP (or Python and Perl). PostgreSQL is often used as a powerful back-end database to run a wide variety of possible websites and online apps. A database designed for commercial transactions. Large and small organisations alike utilise PostgreSQL as their main database to support their apps and products. PostgreSQL is compatible with the most widely used programming languages, such as Python, Java, C#, C/C+, Ruby, JavaScript (Node.js), Perl, Go, and Tcl. PostgreSQL's most notable features are highlighted. User-defined types, inherited tables PostgreSQL shares with other enterprise-class database management systems a sophisticated locking mechanism, Foreign key referential integrity, Views, rules, subquery, Nested transactions (save points), Multi-version concurrency control (MVCC), and Asynchronous replication.

3.5 System Design

The design process has been defined as a multi-step representation of data structure, programme structures, interface features, and procedural detail derived from information needs. Design serves as the basis for all subsequent software engineering and maintenance steps. It is an activity involving decision-making, often of a structural type. Design creates cohesive, well-planned representations of programmes that emphasise the interrelationships between components at the upper level and the logical processes at the lower level. According to the applications and project requirements A good design is one that permits the production of efficient code and whose implementation is as compact as feasible. Design components often comprise functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a comprehensive entity-relationship diagram with a full data dictionary to define the intended software features in detail. These design features are meant to define the software in sufficient depth for qualified programmers to construct it with little extra design input. The core design principle outlines the following procedures for the development of a project: - Figuration - Modularness Software Procedure – Software Architecture – Structural Partitioning – Data Structure System design has two levels: – Logical design - Physical design In logical design, the designer creates a specification of the system's primary characteristics that satisfy the goals. The physical design provides the real system design.

3.5.1 Logical Design

Analysis of the model facilitates comprehension of the interaction between the various system components. The analysis model demonstrates to the user precisely how a system will operate. This is a system's first technical representation. The analytical modelling must accomplish three key goals. To provide a foundation for software design development. To explain the user's requirements. Define a set of criteria that can be verified once the programme has been developed. The objective of a use-case diagram is to represent the dynamic nature of a system. However, this description does

not adequately define the objective. Because the objective of the other four diagrams (activity, sequence, cooperation, and State chart) is same. Therefore, we shall examine the goal that distinguishes it from the other four pictures. Use case diagrams are used to collect the internal and external needs of a system. These criteria are largely design needs. Consequently, when a system is studied to determine its functionality, use cases are created and actors are identified. Once the first effort is complete, use case diagrams are created to illustrate the external perspective. In summary, use case diagrams provide the following functions: Used to collect system requirements. Used to get an outside perspective on a system. Determine the external and internal influences on the system. Demonstrate the interaction between the needs as actors.

3.6 Input Design

Input design is the process of transforming user-generated data into a computer-based representation. The input-handling design considerations govern how data are received for computer processing. Input design is a component of system design that requires thorough consideration and entails defining the mechanism through which actions are to be performed. The collecting of input data is the most costly aspect of the system in terms of equipment and personnel. Entry design involves the acceptance of data for computer processing and the input of data into the system by mapping using map support or linkages. Inaccurate input data is the leading source of data processing problems. The input design process involves the following elements.The first stage in input design is determining which data will be entered. JSON has the same advantages as XML for transferring data in a heterogeneous context, such as the following, when used for data entry on the 'Tekact Platform'. JSON is self-explanatory. In certain instances, the syntax and hierarchical structure of JSON strings might be misconstrued by programmes that are unfamiliar with the expected contents. JSON is a textual format. This makes it appropriate and secure for transmission between platforms and operating systems that do not easily exchange more sophisticated document formats. As text, JSON may be viewed and altered with ease in basic editors. JSON is condensed. A typical JSON string is about two-thirds the size of the equivalent XML string. JSON is simple to learn, simple to read, and simple to comprehend.

The input design's aims are:

1. regulating amount

2. Avoiding delay.

3. Avoiding errors.

4. Avoiding additional steps

All text fields on this 'Tekact Platform' are verified. If a required field is left blank, the notice will be shown.

• The input design is created such that the interface between the user and system is as efficient as possible.

• Measures have been made to reduce user input.

• Additional stages are removed and the procedure is simplified.

3.7 Output Design

Output refers to the system's produced outcomes and data. Determines information to be presented, decides layout and output medium, arranges information presentation in an agreed manner, and decides how output will be sent to intended recipients. The structure and location of column headers and pagination are defined. The output design plays a significant role in supplying the needed format to the user. The output's primary purpose is to transmit information, hence its arrangement and design are carefully considered. Information must be properly tailored to the user's requirements. Standards for printed output recommend naming or titling each Output, providing an example of the output layout, and defining the technique for ensuring the correctness of output data.

Consider the output devices based on their compatibility with the system, reaction time requirements, and desired print quality. Attention is paid to correct identification and language, readability and usability, composition and layout, data item order, and clarity of instructions while designing the output form. A properly formatted form with well stated captions should be self-explanatory. An organization's form must be centralised

for effective processing. The most essential and direct information source for the user is computer output. Output design is a process involving the design of required outputs in the form of reports that should be provided to users based on their needs. Efficient, comprehensible output design should enhance the interaction between the system and the user and facilitate decision making. Since the reports are referred to by management in order to make choices and draw conclusions, they must be prepared with great care, and the information must be concise, descriptive, and easily understood by the user. Therefore, while developing output, the following should be considered:

• Determine which information will be shown.

• Arrange the presentation of information in a way that is acceptable.

• Determine how to allocate output to anticipated revenues.

• Depending on the output's nature and intended future usage, they are presented on the monitor for immediate use and for acquiring a hard copy.

Effective and intelligent output design should enhance the relationship between the system and the user and aid in decision making. Thus, the system may be shown or printed according to the user's preference. A quality output is one that satisfies the needs of the end user and delivers the information in a style that is clear, simple to read, and aesthetically pleasing. A lot of factors, like who gets the output, under what conditions the output is received, etc., must be examined before deciding on an effective way of presentation and format.

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. Efforts to seamlessly integrate such options into adaptive business process management have already found their way into business practice. Furthermore, splitting large applications into individual microservices provides the next degree of independence to agile teams, which supports scaling agile methods. Each team may work on different microservices and develops user stories that affect only their microservices.

4.1 Testing methods

Unit testing is a software development procedure in which the smallest testable components of an application, known as units, are individually and independently inspected for correct operation. Unit testing is an essential step in the software development process because, if performed correctly, it can assist identify code defects that may be more difficult to discover in later testing phases.

Unit testing is a component of test-driven development (TDD), a pragmatic methodology that takes a methodical approach to constructing a product through continuous testing and

refinement.

This testing approach is also the first stage of software testing, performed before to methods such as integration testing. Typically, unit tests are isolated to guarantee that a unit does not depend on external code or functions. Testing can be performed manually, but it is typically automated. Mockito is a well-known open source framework for simulating software test objects. Mockito significantly simplifies the creation of tests for classes with external dependencies. A fake object is a sham implementation of a class or interface. It permits defining the output of certain method calls. They often log interactions with the system, which can be validated by testing.

Typically, a unit test consists of three phases: planning, cases and scripting, and the unit test itself. In the initial phase, the unit test is reviewed and prepared. Following the creation of test cases and scripts, the code is then tested. Test-driven development necessitates that developers first create unit tests that fail. The developers then write code and restructure the application till the test is passed. TDD often results in a code base that is explicit and predictable.

4.1.2 Integration Testing

Integration testing is the second phase of the software testing procedure, which follows unit testing. In this testing, units or individual software components are tested collectively. The objective of integration testing is to identify flaws during the interaction between integrated components or units.

Unit testing employs modules for testing purposes, while integration testing combines and tests these modules. Multiple coders or programmers contribute to the development of the Software's various software components. The purpose of integration testing is to verify proper connectivity between all modules.

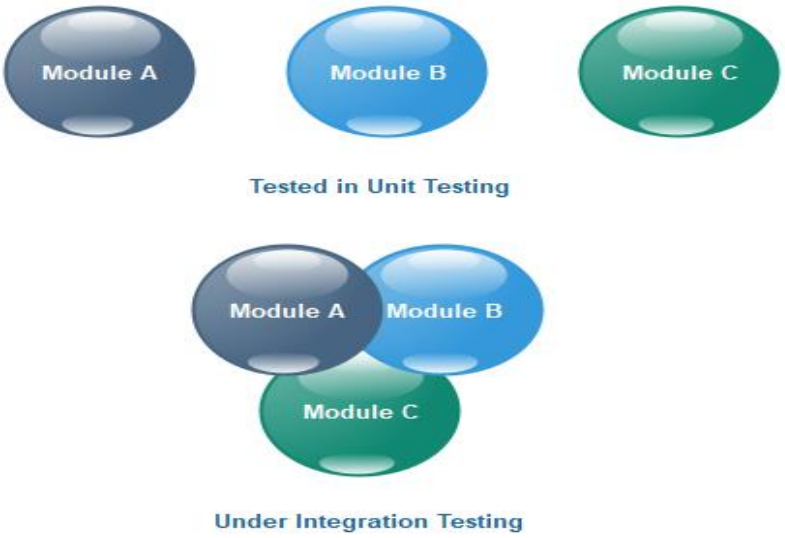


Figure 4.1.2: Modules Testing

4.2 Output Screens and Results

1.Tenant Controller

Admin should be able to create, read, update and delete (soft delete) Tenants. API should be available to retrieve all the tenants in the database.

1.1 Tenant Address Controller

As a Super Admin, it should be able to create, read, update and delete (soft delete) address/contacts to Tenant. API should be available to retrieve all the addresses/contacts in the database

1.2 Tenant Contact Controller

As a Super Admin, admin must be able to create, read, update and delete (soft delete) address/contacts to Tenant. API should be available to retrieve all the addresses/contacts in the database

2.Notify Controller

This module deals with creating Apis for generating notifications over email.

Chapter 5
CONCLUSION

The project sought to do things like offering APIs for building apps, techniques to assemble apps using those APIs, simple integration via REST APIs, workflow management, and offering a functioning app. With the help of these APIs, users will be able to create apps that can function across several domains. a system with many tenants. It employs a software engineering methodology that emphasises breaking down an application into discrete components with clear interfaces. The system's necessity has been determined in an initial effort. A thorough research that is user-friendly and simple to use has been created to satisfy user needs. This particular system has been attractively built so that even a user with little technical expertise might utilise it without difficulty. This system is quite helpful. This is a microservices-based product, where each microservice deals with issues unique to a certain domain. Through the integration of those many microservices, users will build apps. These days, many businesses are leading technological advancements, and developing reusable services and APIs speeds up the time it takes for new goods to reach the market. Using the best microservices principles to provide scale, performance, and resilience, this microservices-based project shows how several services can operate independently.

5.1 Future Enhancement

The system is designed in such a way that addition of new modules can be done without much difficulty. The reconstruction of the system will increase the flexibility of the system. The system has been developed as versatile and user friendly as possible keeping in mind the advanced features in this technology.

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

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

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

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

[5].Jishnu Saurav Mittapalli and Menaka Pushpa Arthu,“Survey on Template Engines in Java”,School of Computer Science and Engineering, Vellore Institute of Technology, Chennai Campus,2020 India 600 127

[6].Lei, Kai, Yining Ma, 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 .

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

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

APPENDIX

Screenshot

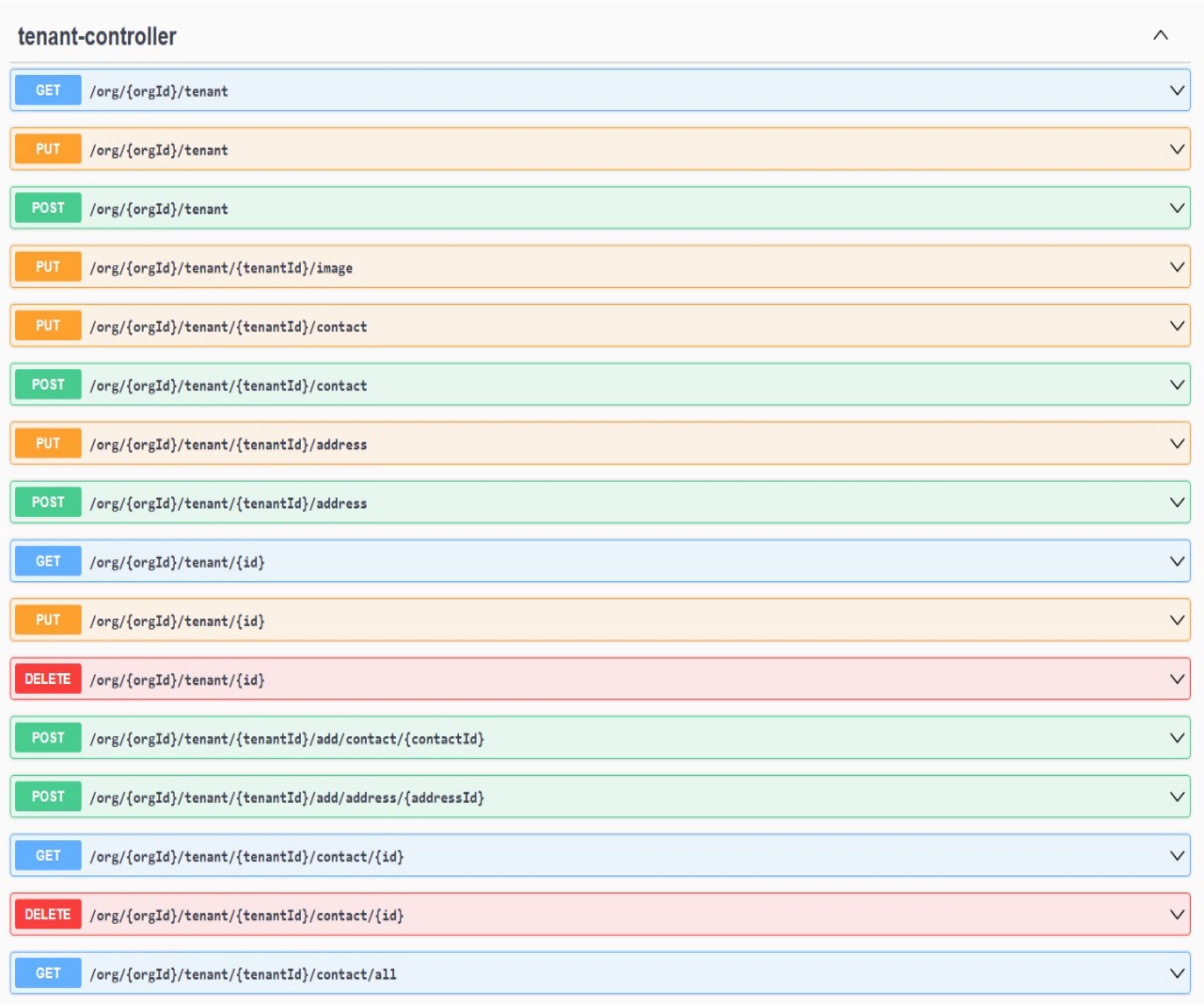


Fig A. 1 Tenant controller

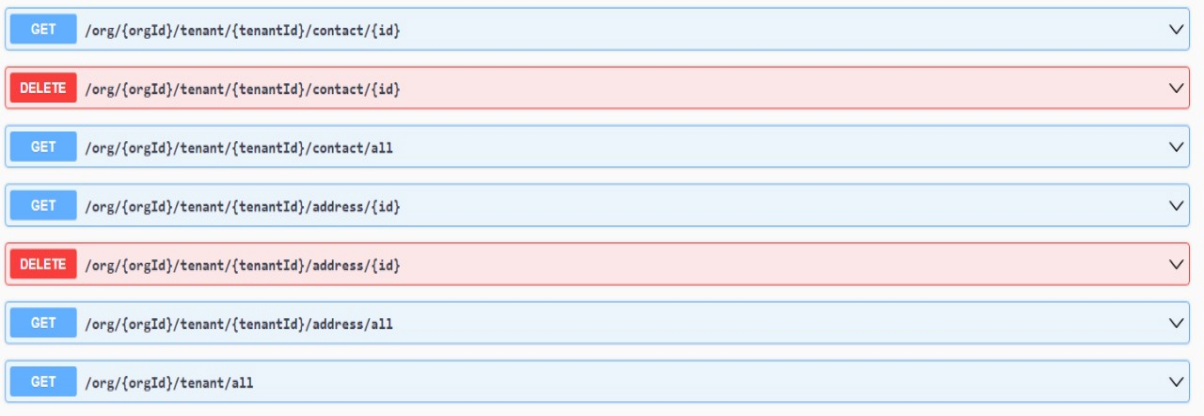


Fig A. 2 Tenant Address, Contact Controller

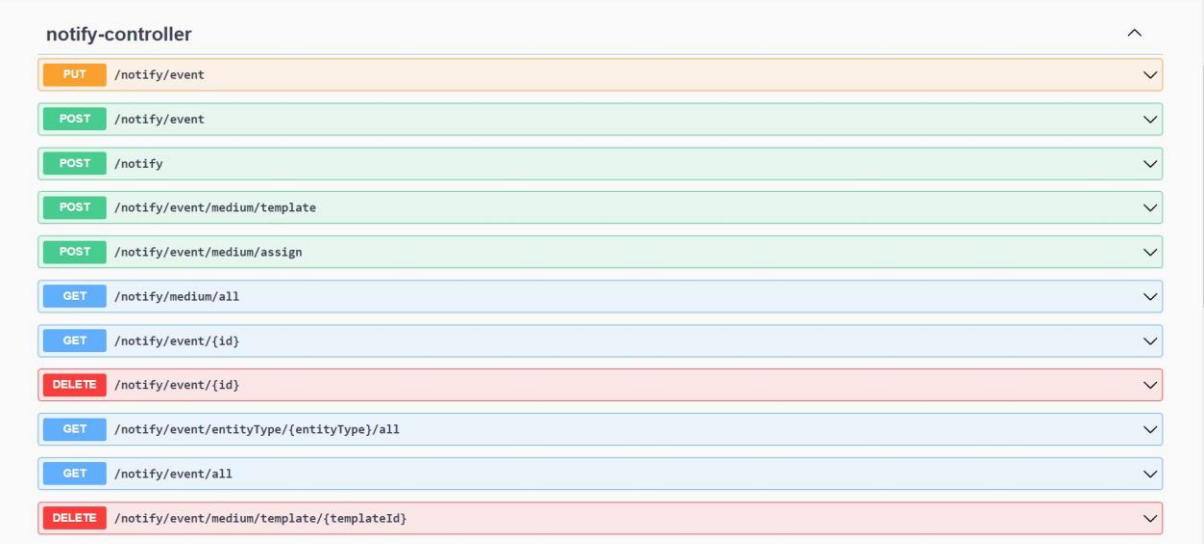
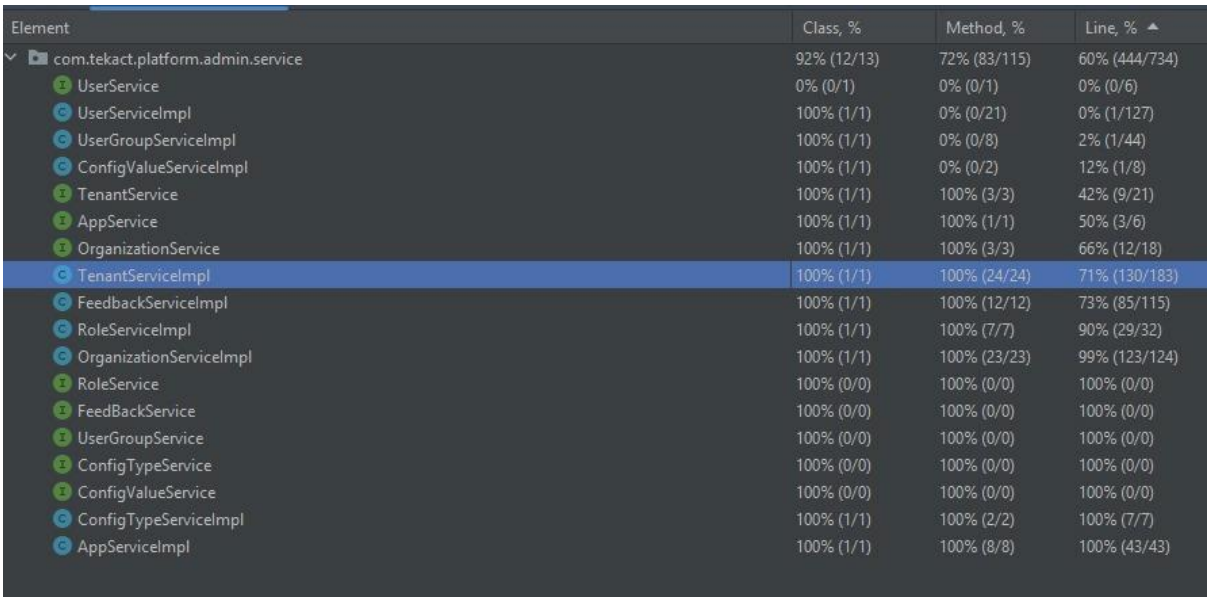
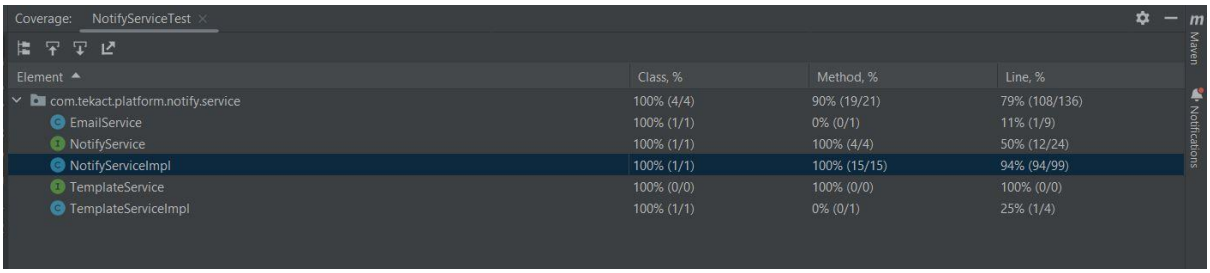


Fig A. 3 Notify Controller



Element	Class, %	Method, %	Line, %
com.tekact.platform.admin.service	92% (12/13)	72% (83/115)	60% (444/734)
UserService	0% (0/1)	0% (0/1)	0% (0/6)
UserServiceImpl	100% (1/1)	0% (0/21)	0% (1/127)
UserGroupServiceImpl	100% (1/1)	0% (0/8)	2% (1/44)
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)
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)
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)

Fig A. 4 Tenant Test Coverage



Coverage: NotifyServiceTest

Element	Class, %	Method, %	Line, %
com.tekact.platform.notify.service	100% (4/4)	90% (19/21)	79% (108/136)
EmailService	100% (1/1)	0% (0/1)	11% (1/9)
NotifyService	100% (1/1)	100% (4/4)	50% (12/24)
NotifyServiceImpl	100% (1/1)	100% (15/15)	94% (94/99)
TemplateService	100% (0/0)	100% (0/0)	100% (0/0)
TemplateServiceImpl	100% (1/1)	0% (0/1)	25% (1/4)

Fig A. 5 Notify Test Coverage