

Digital Customer Profiles

PREPARED BY

Sanket Sharma (2018BTechCSE116)

FACULTY GUIDE

Mr. Gaurav Raj



Department of Computer Science & Engineering

Institute of Engineering and Technology (IET)

JK Lakshmipat University, Jaipur

May 2022

Digital Customer Profiles

Practice School-2

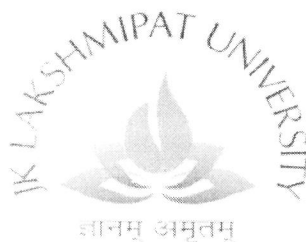
Submitted in partial fulfilment of the requirements
for the degree of
Bachelor of Technology in Computer Science Engineering

By:

Sanket Sharma (2018BTechCSE116)

Faculty Guide:

Mr. Gaurav Raj



Department of Computer Science & Engineering
Institute of Engineering and Technology (IET)
JK Lakshmipat University, Jaipur

May 2022

CERTIFICATE

This is to certify that the Practise School-2 project work entitled “**Digital Customer Profiles**” submitted by **Sanket Sharma (2018BtechCSE116)** towards the partial fulfilment of the requirements for the **Bachelor of Technology in Computer Science Engineering** of JK Lakshmipat University Jaipur is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for Practise School-II examination.



Mr. Gaurav Raj

Assistant Professor,

Department of Computer Science
Engineering

Institute of Engineering & Technology
(IET)

JK Lakshmipat University Jaipur



Dr. Sonal Jain

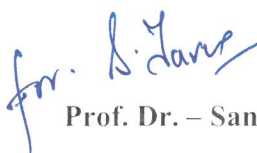
HOD, Co-Ordinator and Professor,

Department of Computer Science
Engineering

Institute of Engineering & Technology
(IET)

JK Lakshmipat University Jaipur

Date of Submission: 13th May 2022



Prof. Dr. – Sanjay Goel

Director

Institute of Engineering & Technology

JK Lakshmipat University

ACKNOWLEDGEMENTS

The internship opportunity I had with ZopSmart Technology Pvt. Ltd. was a great chance for learning and professional development. Therefore, I consider myself fortunate individual as I was provided with an opportunity to be part of ZopSmart. I am also grateful for having a chance to meet so many wonderful people and professionals who led me through this internship period.

I perceive as this opportunity as a big milestone in my career development. I have taken efforts in completing this internship. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to express heartfelt gratitude to **Prof Dr.- Sanjay Goel**, Director, Institute of Engineering & Technology for providing this opportunity to me. Also, I would like to thank my faculty guide **Mr. Gaurav Raj** for his constant support and guidance.

I am highly indebted to **ZopSmart Technology Pvt Ltd.** for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express our gratitude towards our parents & the members of ZopSmart Technology Pvt Ltd. for their kind co-operation and encouragement which helped me in the completion of this project and internship. I would like to express my gratitude and thanks to **Mr. Vikash Kumar, Mr. Ujjawal Misra Mr. Anshul Katare and Mr. Divyansh Singla** industry persons for giving me such attention and time.



Sincerely yours,
Sanket Sharma

(2018BtechCSE116)

ABSTRACT

It is a need for any online retail company to provide their users with a seamless buying experience to the users who come to their platform for any kind of purchase. The first step that the users take while buying anything online is create an account that they can use to proceed buying what they require and that is what the role of Digital Customer Profile team's function is. Digital Customer Profile team or DCP team is the team responsible for letting users create their accounts in the client's websites.

Digital Customer Profiles is a client service handled by a set of onshore and offshore employees to ensure the customers can handle operations on their digital accounts which they use for shopping over the different websites owned by the client. This includes creation of digital accounts, updating of user details like name, addresses, mobile number and other fields, deletion of an account and management of other internal features of the team to ensure all these features are provided to the user in a smooth fashion.

To achieve this goal, the digital accounts team works with multiple client teams to get details from them, provide details to them and ensure that all the work that is being done is in sync with every team so that the production code doesn't face any issues. Every feature of the service is well coordinated between the teams and is released only after proper testing and cutover is done over the stage environment after which the features are deployed on Kubernetes clusters.

LIST OF FIGURES

Figure	Page No
Figure 2.1 Hierarchy of services under DCP	4
Figure 2.2 Development flow for features	7
Fig 2.3: Deployment flow for features	13

LIST OF TABLES

Table	Page No
Table 4.1 Basic get by user id test	18
Table 4.2 Get by user id test	19
Table 4.3 Exists test	20

CONTENTS

<i>CERTIFICATE</i>	<i>iii</i>
<i>ACKNOWLEDGEMENT</i>	<i>iv</i>
<i>ABSTRACT</i>	<i>v</i>
<i>LIST OF FIGURES</i>	<i>vi</i>
<i>LIST OF TABLES</i>	<i>vii</i>
CHAPTER 1: INTRODUCTION	1-2
1.1 About the organization	1
1.2 Digital Customer Profiles	2
CHAPTER 2: ABOUT THE SERVICES	9-17
2.1 Problem Statement	3
2.2 Services	3-4
2.2.1 The BFF services	3
2.2.2 The CPS	3
2.2.3 The DC service	3-4
2.3 Responsibilities as a developer	5-6
2.3.1 Code Responsibilities	5
2.3.1.1 Understanding the code base	5
2.3.1.2 Interaction with different services	5
2.3.1.3 Test complete codes	5
2.3.1.4 Logging	5
2.3.2 Other Responsibilities	6
2.3.2.1 Deployments	6
2.3.2.2 Bug fixes and communications	6
2.3.2.3 On call support	6
2.4 The lifecycle of a feature	6-7
2.4.1 The Requirement	6
2.4.2 Spike	6
2.4.3 Story	7
2.4.4 Cutover	7
2.4.5 Deployment to production	7
2.5 Tools and technologies	8-15
2.5.1 Language and Technologies	8
2.5.2 Code Development	9
2.5.3 Logging	9

2.5.4 Version Control	10
2.5.5 Containerization	11
2.5.6 Testing	11
2.5.7 Database	13
2.5.8 Deployments	13-14
2.5.9 Support	15
CHAPTER 3: SOFTWARE DESIGN	16-17
3.1 Design Strategy	16
3.1.1 Function Oriented Design	16
3.2 Feasibility Study	16
3.2.1 Technical Feasibility	16
3.2.2 Time Schedule Feasibility	17
3.2.3 Operational Feasibility	17
3.2.4 Implementation Feasibility	17
CHAPTER 4: CONSTRUCTION	18-19
4.1 General Introduction	18
4.2 Test Scenarios	18
4.2.1 Basic get by user id endpoint	18
4.2.2 Get by user id endpoint	19
4.2.3 Exists Endpoint	19-20
CHAPTER 5: FEATURE WORK AND TASKS PERFORMED	21-23
CHAPTER 6: CONCLUSION	24
REFERENCES	25

CHAPTER 1: INTRODUCTION

1.1 About the organization

ZopSmart Technology is a cutting-edge retail technology company that provides you all the tools to build your e-commerce business. Whether you are a conventional retailer intending to build your omni-channel business or an online-only retailer intending to build your e-commerce business, we have a suite of products that will quickly and easily empower you to achieve your goals. They envision to be the #1 retail technology platform of the world. They have their expertise in the retail domain is built on our team's immense experience in the field of FMCG, retail, supply-chain and logistics. All our experience has been baked into business processes that are embedded in our products. Their solution has been built over 7 years and is handling thousands of transactions every day. Our suite of products are one of the most advanced, stable and scalable solutions available in the world of retail technology. Their solutions include e-commerce, m-commerce, Digital Marketing, Customer Management, e-wallet and Order Management etc.



Using e-Commerce they give customers a world-class shopping experience - Intuitive search to quickly find products. Personalized product listing for ease of ordering. Self-service rescheduling and returns functionality for frictionless order experience. M-Commerce provide your customers a superlative shopping experience on mobile devices - Responsive website that adapts to all screen sizes. Native Android and iOS mobile apps for world-class experience. Persistent cart across devices for convenience. E-Wallet uses integrated wallet to build customer loyalty - Integrated wallet to enable quick and hassle-free checkout. Refunds and cashback to wallet to encourage repeat purchase. Wallet-credit as gift-cards for acquiring of new customers at low cost. Customer Management manage your online as well as offline customers easily and efficiently - Single interface to view all your customers with their contact details. Available details of each customer's purchase history, orders and loyalty points. Option to contact the customer from the interface by phone-call, SMS and e-mail.

1.2 Digital Customer Profiles

Digital Customer Profile is a client project team of onshore and offshore employees who work on the digital profiles created by customers when they visit any of the client's website. This includes creation, deletion and updation of account details and interaction with different services to provide the necessary data to the upstream UI to provide to the users. The role of this service is to provide the users with easy and fast login/signup to enter the shopping phase on the platform and maintain their accounts.

The different subsidiaries that the client have some minor uniqueness into them and these are also handled in the services operated under DCP. Apart from the user roles, it is also a job of the DCP service team to develop and maintain internal services which can be used to access sensitive data by the authorized teams. The team handles the full lifecycle of the creation of different features while communicating with different teams to get and post data to them.

Also, the team holds many projects under them which includes the addition of required features for any new subsidiary that the client acquires, managing the currently running services which are live websites on which customers shop and resolution of bugs if anything goes wrong with any of the services. With that, the team is also involved in optimizations of the services as per the request response times as well as how the requests are transferred over the network. A close eye is kept on how the different services interact with each other to find optimised implementations that have fast response times.

CHAPTER 2: ABOUT THE SERVICE

2.1 PROBLEM STATEMENT

Creating and managing features related to a customer's digital accounts in any platform is a high importance task for any retail organization as minutes of downtime in these features can lead to a loss of huge sums of money as well as potential future loyal customers. So, creation of clean features with low response times and management of these features in case of any bugs on high priority is one of the biggest needs of any ecommerce company.

2.2 SERVICES

The Digital Customer Service team operates over three major services while also having some other projects under it. These services interact with each other in a hierarchical pattern and have distinguished functionalities and implementations in them.

2.2.1 The BFF Service

The first service under this team is Bff Service which consists of Profile-bff and Accounts-bff layer. These are the point of contact between the UI and the backend for create and update operations. This layer ensures that all the information provided by the UI is provided in a correct format with the right authentications. It acts as more of an authentication layer between the UI and the main layer which contains all the business logic.

2.2.2 The CPS

The Customer Profile Service is the service which contains all the business logic of all the features. It is the one layer which interacts with the services handled by different teams to get data from them or post data to them. Here we have the logic related to account creation, updation, deletion, card allotments and user preferences. This layer is currently in a migration mode to a new service and is now converted into two different shim layers one of which handles all the get calls and other handles the write calls. This layer is the downstream layer to the bff layers and has a downstream service called the DC service which directly interacts with the database.

2.2.3 The DC Service

The digital customer service is the service which handles the logic related to the database. It contains all the logic and queries required to put and fetch data from the database. These services use a Cassandra database with multiple schemas and are used according to the necessity of the request that come. Cassandra provides a fast read write solution which is required when handling so many requests coming to these services at the same time.

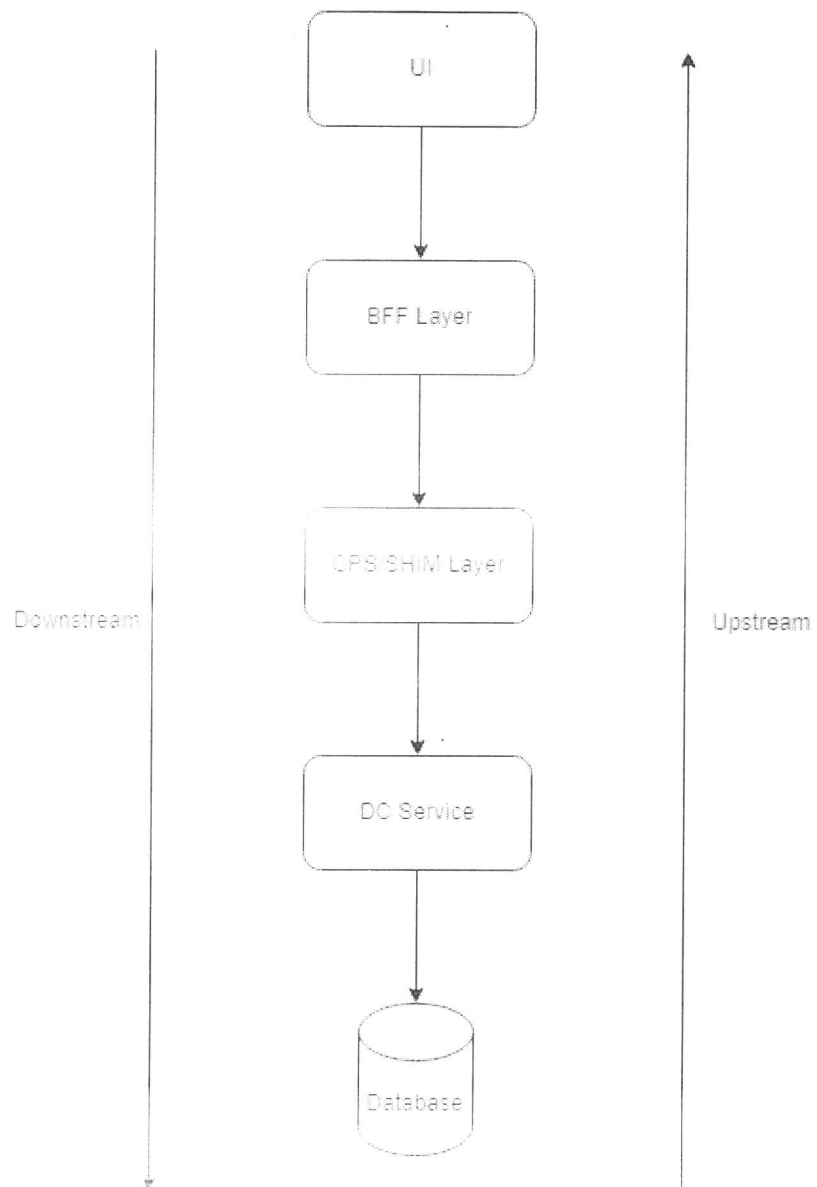


Figure 2.1 : Hierarchy of services under DCP

This diagram shows the hierarchy of interaction of core services under DCP. There are interactions of these services with other services which are controlled and managed by other teams.

2.3 RESPONSIBILITIES AS A DEVELOPER

Working in a team which handles the customer profiles comes with a lot of code development as well as other service-related responsibilities like proper deployment, testing, bug reporting, bug fixing and many other things.

2.3.1 Code responsibilities

2.3.1.1 Understanding the code base

The code base that we are working on has been worked on upon since a long time and there are a lot of people who have and are interacting with the code continuously. Proper understanding of what the code does, which services does it interact with and being able to reuse an already written code to reduce the time taken in development and complexity of code is an important task.

2.3.1.2 Interaction with different services

There are different services handled by the DCP team. These services follow a proper hierarchy of development and have different authorized access to certain other services. Understanding of the interactions these services have within themselves and with the services of other teams becomes important as any change in the existing code comes with a possibility of something going wrong with the other services. Keeping the code safe is a huge responsibility to ensure saving of time and resources in case of bugs occurring in services.

2.3.1.3 Test complete codes

As a developer, we are expected to write test complete codes to ensure testing of every feature that is created. This includes writing unit test cases and integration test cases for the features which are developed. This ensures that the services are building properly without bugs and also helps the other developers in the team in understanding what a particular piece of code is expected to do.

2.3.1.4 Logging

Logging is a very important feature/necessity that the code with so many services interacting with each other requires. Without proper logging and log management, it would be impossible for the developers to resolve any bugs that occur in any service. Logging helps in pinpointing which services are causing the issue and if it's our service, we try to fix the issue and release a hotfix if required. If the issue is due to some other team's service, we can pass the ticket to them, and this can be achieved only by tracking the api requests using logs.

2.3.2 Other Responsibilities

2.3.2.1 Deployments

Developers need to have three environments – dev, test and stage to test all the changes/ new features that they have created. Deployment can be done on these environments which creates a docker image of the current service which is then deployed in a Kubernetes cluster. For deployments in the production environment a release candidate is created for every feature which is tested and deployed in a deployment time window which is fixed for every day.

2.3.2.2 Bug Fixes and Communications

While developing any feature, we have to keep in mind the possible bugs that it will produce, the severity of the bug and the potential solutions to prevent or reduce it. In case any bug is reported, the team tries to get to the root of it and communicate a possible fix.

2.3.2.3 On call support

Two of the developers from the team must provide the on-call support as a primary and secondary point of contact. It's a 12-hour timeline during which if any major incident is reported by the operational teams, the primary must investigate that bug and join any bridge meetings if necessary. If primary is not available, the call goes to secondary, and the on-call support keeps rotating developers on a weekly basis.

2.4 The lifecycle of a feature

This section talks about the general idea of a lifecycle that every feature under development goes through with some differences as per the requirements of different features. A generalized roadmap of any feature can be defined as below –

2.4.1 The requirement

The team works in a Kanban lean methodology in which we have quarters defined for the team. For each quarter, a certain set of features is decided to be implemented by the team while doing all the other developer duties. This is where the groundwork like the requirement for this feature and the idea behind it is laid down.

2.4.2 Spike

The developers are required to pick cards based on their availability and priority of tasks. When a feature is taken up, a spike is started for that feature. A spike consists of figuring out possible efficient ways of implementing a feature and making sequence diagrams for that implementation. These diagrams and ideas are then discussed with the teams whose services will be called by the feature (if any) and on approval, a flow of implementation is decided, and the spike is closed.

2.4.3 Story

After the successful completion of a spike ticket, a story is raised for that feature. A story ticket requires the developer to develop the feature according to the fashion decided in the spike for it, test for the correct responses, write test cases for it and pass it on to the tester. The tester tests the ticket and informs of changes if any are required. If not, the ticket is considered ready for deployment.

2.4.4 Cutover

From there, there can be different steps based on what the feature is. If the feature requires a proxy cutover, a card is generated for that. For this, the team makes changes to the code managed by a different team and generates a pull request which on proper testing is approved by the team.

2.4.5 Deployment to production

After the cutover (if required) a release candidate is generated for that feature after taking proper approvals and the feature is deployed on the production site. Proper testing is done for that feature after that and if something goes wrong, the feature is rolled back. Otherwise the feature stays in production.

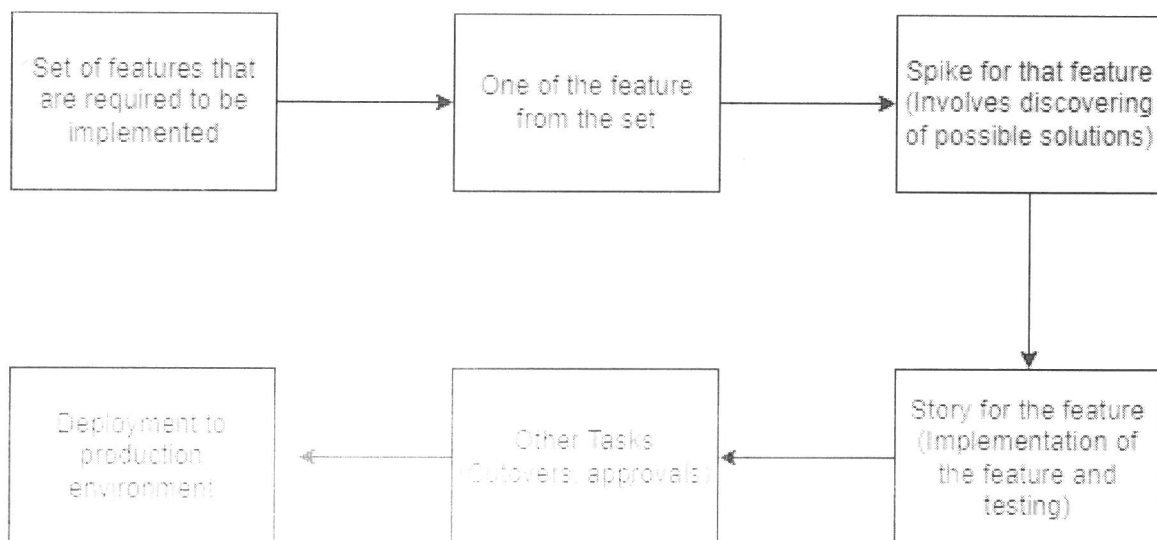


Figure 2.2: Development flow for features

2.5 Tools and Technologies

To fulfil the developer duties, we are assisted with some tools and technologies which makes the work that we do possible and efficient. This section talks about the different tools and technologies that we use in the development of a whole feature cycle. The tools providing aid in similar fields are grouped together.

2.5.1 Language and technologies

This section talks about the language that is used from programming the features of the service and also the different distinct frameworks and libraries that are used for production, management and documentation of these features.

Java

In this project we use java as the coding language. Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.



Spring framework and Springboot



The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform.

A key element of Spring is infrastructural support at the application level: Spring focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments.

Spring Boot is an open-source Java-based framework used to create a micro Service using the spring framework. It makes development of features and API faster by providing annotations that perform configuration tasks in an efficient and fast manner.

Hystrix

Hystrix is a library designed by Netflix to control points of access to remote systems, services and 3rd party libraries, providing greater tolerance of latency and failure. Hystrix isolates the points of access between the services, stops cascading failures across them and provides the fallback options.



Swagger



Swagger is a suite of API developer tools from SmartBear Software and a former specification upon which the OpenAPI Specification is based.

Swagger is used for API documentation. Swagger allows you to describe the structure of your APIs so that machines can read them. The ability of APIs to describe their own structure is the root of all awesomeness in Swagger.

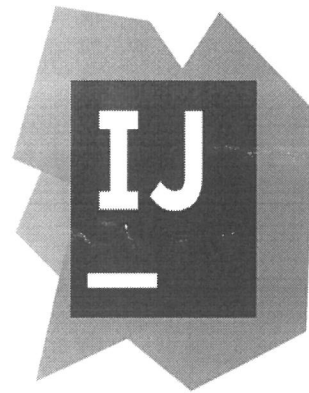
2.5.2 Code development

This section talks about the tools that are used to write and debug the code.

IntelliJ Idea Ultimate

IntelliJ IDEA is an integrated development environment written in Java for developing computer software. It is developed by JetBrains and is available as an Apache 2 Licensed community edition, and in a proprietary commercial edition. Both can be used for commercial development.

It provides good support for development by giving great tools for Gradle implementation, building and debugging application with also having features like navigation through libraries in java, finding usages of specific codes and checking test coverage of codes. The debug tool of this IDE is super powerful and provides a lot of information during debugging which makes the bug finding and fixing process fast and convenient.



2.5.3 Logging

This section talks about the different tools used to log the events and api requests in the services. These tools are not limited to just logging but also provide clear visualization of the requests and responses based on the filters that are selected in them, tracking a request from the initial to the ending point and bringing out the point of failures and also checking the cpu and memory usages that the requests are taking while their course of existence.

Elastic



Elastic Logging is a service that scales for you, automatically ingesting data of any source, size and format without the need for further configuration by the end user.

Elastic Stack provides centralized logging in a low-cost, scalable, cloud-friendly manner. Its user interface streamlines data analysis so you can spend your time gleaming insights from your data instead of fighting with a clunky interface.

Kibana is a proprietary data visualization dashboard software for Elasticsearch, whose free and open source successor in OpenSearch is OpenSearch Dashboards.

Dynatrace

Dynatrace enables monitoring of your entire infrastructure including your hosts, processes, and network. You can perform log monitoring and view information such as the total traffic of your network, the CPU usage of your hosts, the response time of your processes, and more.

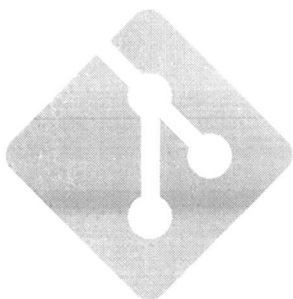
With Log Monitoring as a part of the Dynatrace platform, you gain direct access to the log content of all your mission-critical processes. You can create custom log metrics for smarter and faster troubleshooting. You will be able to understand log data in the context of your full stack, including real user impacts.



2.5.4 Version control

This section talks about the version control and the platform used for it. Here are the tools which helps in developing and modification of code on bigger scale while keeping the integrity of code intact and keeping the code secure.

Git



Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is used for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

GitHub

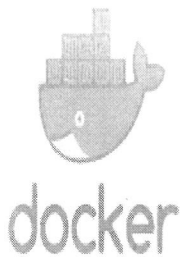
GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. It offers the distributed version control and source code management functionality of Git, plus its own features.



2.5.5 Containerization

This section contains the tools that we use to containerise the application which helps in running the application in different environments without the need of manually updating the system and storing these containerised images in the artifactory.

Docker



Docker is a set of platforms as a service product that use OS-level virtualization to deliver software in packages called containers. The service has both free and premium tiers. The software that hosts the containers is called Docker Engine. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment.

Jfrog

JFrog Pipelines is an automation solution for building, testing, and deploying software as part of your CI / CD pipeline. It provides end-to-end orchestration and optimization of all key processes of your DevOps pipeline. It is used as an artifact management tool by the team to store the containerised applications and providing them whenever needed.



2.5.6 Testing

This section contains the tools that are used by us developers to write the unit and integration test cases for every feature that we write. It also contains tool (postman) which is used to test the API responses for the API's that we develop while providing additional features to make testing them fast and smooth.

JUnit



JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development and is one of a family of unit testing frameworks which is collectively known as xUnit that originated with SUnit. JUnit is linked as a JAR at compile-time.

Mockito

Mockito is an open-source testing framework for Java released under the MIT License. The framework allows the creation of test double objects in automated unit tests for the purpose of test-driven development or behaviour-driven development.



Postman



Postman is an API client that makes it easy for developers to create, share, test and document APIs. This is done by allowing users to create and save simple and complex HTTP/s requests, as well as read their responses. The result - more efficient and less tedious work.

Postman is very convenient when it comes to executing APIs. Once you've entered and saved them, you can simply use them over and over again, without having to remember the exact endpoint, headers, API keys, etc.

Jmeter

JMeter is a test tool from Apache used to analyze and measure the performance of applications, different software services and products. It is open source software entirely written in Java, used to test both web and FTP applications as long as the system supports a Java Virtual Machine (JVM).



2.5.7 Database

This section contains the database that we use to store the data provided by different API's as a response and provide the data whenever it's needed.

Cassandra

Cassandra is a free and open-source, distributed, wide-column store, NoSQL database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure.



2.5.8 Deployments

This section contains all the tools and technologies that are used in the deployment process. The flow of these deployments is as follows:

After a RC (Release candidate) is created for a feature, a Github workflow for build and dockerizing runs which contains configuration to first build the project, then dockerize the project. This docker image is then stored in a jfrog artifactory.

When a github workflow for deployment is run, a trigger calls out harness to deploy the image stored on the jfrog artifactory on a kubernetes cluster which is already configured for the deployment environment it is run for. For different environments, there are different workflows which call different triggers to run deployment on different environments.

During deployment, harness does a blue/green deployment which means that there is no downtime of services and as soon as the deployment is done the previous cluster shuts off and new one starts resulting in zero down time. Harness also makes sure that if the deployment goes wrong, its automatically rolled back to the previous stable version.

These Kubernetes clusters are managed with the help of rancher which makes the job of developing and managing these clusters seamless. A script is also written to scale down the running instances of a deployment cluster after the other cluster is up and running.

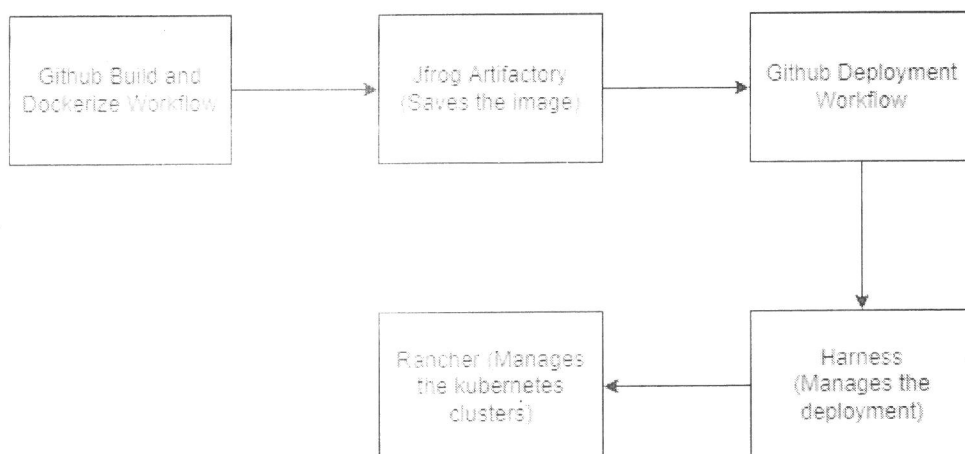
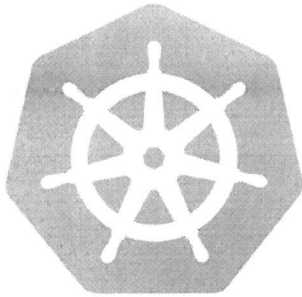


Fig 2.3: Deployment flow for features

Kubernetes

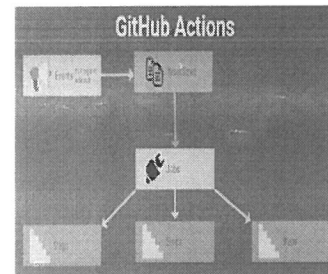


Kubernetes is an open-source container orchestration system for automating software deployment, scaling, and management. Google originally designed Kubernetes, but the Cloud Native Computing Foundation now maintains the project.

Kubernetes, often abbreviated as “K8s”, orchestrates containerized applications to run on a cluster of hosts. The K8s system automates the deployment and management of cloud native applications using on-premises infrastructure or public cloud platforms.

Github Workflows

A workflow is a configurable automated process made up of one or more jobs. You must create a YAML file to define your workflow configuration. A github workflow can be configured to do certain jobs that are required in an automated manner thus reducing the need of doing similar tasks again and again and increasing work efficiency. They can be configured to do tasks such as building applications, containerizing them and deploying them on particular environments.



Harness



Harness is the industry’s first Software Delivery Platform to use AI to simplify your DevOps processes - CI, CD, Feature Flags, Cloud Costs, and much more. Harness CD is a Self-Service Continuous Delivery module that enables engineers to deploy on-demand without scripts, plugins, version dependencies, toil and downtime.

Rancher

Rancher is an open-source software platform that enables organizations to run containers in production. With Rancher, organizations no longer have to build a container services platform from scratch using a distinct set of open-source technologies.



2.5.9 Support

XMatters



xMatters is a service reliability platform that helps DevOps, SRES, and operations teams automate workflows, ensure infrastructure and applications are always working, and rapidly deliver products at scale.

CHAPTER 3: SOFTWARE DESIGN

3.1 DESIGN STRATEGY

Software design is the process of developing software to solve one or more problems. It converts user and client requirements into a format that programmers can use for development and implementation.

Problem addressing and software solution planning are common tasks in software design. This encompasses component and algorithm design at the low level, as well as high-level architecture design.

Furthermore, software design shifts the focus from the problem domain to the solution drama. It seeks to describe how to meet the requirements in Software Requirement Specifications (SRS).

Software Design Strategy is a discipline that assists businesses in determining what to build, why to do it, and how to innovate contextually in the short and long term. This process involves strategic design and interplay between design and business strategy.

The software design strategy is mainly about organizing design activities during the course of design.

3.1.1 Function oriented design

The system in a function-oriented architecture is made up of numerous smaller sub-systems called functions. These functions can complete important system tasks. All functionalities are viewed from the top of the system.

The system is developed according to the functionality it provides in a function-oriented design. The system is initially inspected broadly, and then each of its functions is examined to determine which sub-functions are responsible for carrying out that functionality.

There is a centralised function that consists of various functions in this type of architecture. The programme follows the same pattern.

The feature development and service management in DCP is also performed in the similar manner where for every feature that is being developed, functions related to particular functionality are made and those can be called and reused whenever necessary. It gives high reusability of code which makes development times lower.

3.2 Feasibility Study

3.2.1 Technical Feasibility

A technical feasibility study evaluates the details of how the developers intend to build a system or solution. Technical analysis evaluates technical merits of the system and at the same time collects additional information about performances, reliability, maintainability, and productivity. In this section feasibility study of the project is analysed.

Basically, this study is to ensure that proposed system software is not a loss or burden to the user. It assesses the practicality of the proposed project. For this study, core knowledge of the system is required.

3.2.2 Time Schedule Feasibility

The Project has simple working and basic requirement can be satisfied with the allotted period.

3.2.3 Operational Feasibility

The proposed system will increase the operational efficiency of all users. Therefore, the throughput of the system and reduction of time is as desired.

This system will ensure and satisfy the requirements identified by the requirement analysis phase of the system.

The API will be user-friendly and will work with any frontend system for the same definition i.e., it will be very useful and easy to operate.

3.2.4 Implementation Feasibility

User can directly access API with the Postman or any API testing software or also can create the frontend and API can be useful for the same.

Implementation will result in all the important information about all the available professionals and customers can hire any of them for their personal activities/needs.

CHAPTER 4: TESTING

4.1 GENERAL INTRODUCTION

Software testing is the process of verifying the correctness of software by considering all attributes and evaluating the implementation of software components in order to detect bugs, errors, or flaws. Readability, scalability, portability, reusability, and usability are all software attributes.

Software testing provides a stand-alone view and objectivity of the software while also ensuring its accuracy. It entails putting all parts through the appropriate procedures to ensure that they meet the set requirements. The method also provides information about the product's nature to the customer.

The manual testing technique is followed to test the services. The process of testing software without using any automation tool is termed as manual testing. There are three types of manual testing namely white box testing, black box testing and grey box testing. The manual testing is done using functional unit testing approach which is a sub type of black box testing.

Functional testing emphasizes upon validation of the functionalities of the software by checking whether it is working as per the requirements specified. Each feature is tested by giving the input, determining the output and verifying the actual output with the expected output.

4.2 TEST SCENARIO

4.2.1 Basic get by user id endpoint

- a) Test case 1: Id exists in EC
- b) Test case 2: Id exists in DC not in EC
- c) Test case 3: Id does not exist

Test Case Number	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
TC1	Check EC user	User-Id: ec-user-id (undisclosable) Banner: banner-x	Should get result from EC	As expected	Pass
TC2	Check DC user	User-Id: dc-user-id (undisclosable) Banner: banner-x	Should get result from DC	As expected	Pass
TC3	Check user not present	User-Id: non-existing-user-id (undisclosable) Banner: banner-x	Give out exception	As expected	Pass

Table 4.1 Basic get by user id test

4.2.2 Get by user id endpoint

- a) Test case 1: Id exists in EC
- b) Test case 2: Id exists in DC not in EC
- c) Test case 3: Id does not exist

Test Case Number	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
TC1	Check EC user	User-Id: ec-user-id (undisclosable) Banner: banner-x	Should get result from EC	As expected	Pass
TC2	Check DC user	User-Id: dc-user-id (undisclosable) Banner: banner-x	Should get result from DC	As expected	Pass
TC3	Check user not present	User-Id: non-existing-user-id (undisclosable) Banner: banner-x	Give out exception	As expected	Pass

Table 4.2 Get by user id test

4.2.3 Exists endpoint

- a) Test case 1: Email id exists in EC
- b) Test case 2: Email id exists in DC not in EC
- c) Test case 3: Email id does not exist
- d) Test case 4: Email id exists but not in particular banner

Test Case Number	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
TC1	Check EC user and in banner exists	Email: ec-email (undisclosable) Banner: banner-x	True	As expected	Pass
TC2	Check DC user and in banner exists	User-Id: dc-email (undisclosable) Banner: banner-x	True	As expected	Pass
TC3	Check user not present because id not present	User-Id: non-existing-user-id (undisclosable) Banner: banner-x	False	As expected	Pass
TC4	Check user not present because user not present in banner	User-Id: existing-user-id (undisclosable) Banner: banner-y (non-existence of id)	False	As expected	Pass

Table 4.3 Exists test

CHAPTER 5: FEATURE WORK AND TASKS PERFORMED

After completing a one-month training on topics like git, java, docker, postgres, JDBC and Spring boot I was moved to a team working for a client. During working with this team, I performed several tasks while learning all the different tools and their proper usages. This section will talk about some of the major tasks I performed as a developer in the team.

Task 1: App scan document for endpoints

In order for vulnerability check of the endpoints that are created, an app scan document is required to be submitted to an ethical hacking team which hits the API for any safety issues that it might have. In this app scan document, we are required to submit some sample request bodies and responses with the possible success and error outcomes that the endpoint might give based on the different scenarios. I completed making the app scan for a pre existing endpoint which became the first card that I completed in my Internship.

Task 2: Migration of a basic get endpoint to another service

The CPS service is undergoing a migration into a new service which requires the migration of the endpoints from our services to a service handled by another team called EC. This migration means that if that particular endpoint is called, the call will first go to the EC databases. If the data is not available in EC we will look for it in the DC service and if the data is present nowhere, the API will give a proper error response. For this migration, I was provided with the flow diagram and I had to figure out a proper way of mapping all the necessary fields to fulfil the previous contract that the API had. After figuring that out, I implemented the changes in the code and made the changes that were asked by the testers while testing of the card. After that, the card/feature was ready to be deployed in production.

Task 3: Proxy Cutover for the get endpoint

After an endpoint is development and test complete, it is ready to move to the production environment. For that the current traffic is needed to be cutover to this endpoint. In order to do so, we do a proxy cutover in which we have to edit configuration files which are handled by a different proxy team and get the changes approved by them to move the traffic to the new endpoints once they are in the production environment. I made those configuration changes in the config files of the proxy team repo to do a cutover of the endpoint I migrated in task 2.

Task 4: Spike for migration of exists endpoint

The next task was to migrate an exists endpoint which returned true or false if a user exists under a particular email and some other conditions. For this endpoint, I started a spike ticket in which I had to come up with an optimized solution to achieve the migration while keeping the response times low and create a sequence diagram for the implementation flow which I was thinking to follow. During this, had to keep in mind all the different services that I would interact with, analyse the responses that the other team's endpoints provided and communicate with those teams about our team calling their endpoints. After discussion with the concerning teams and making the changes in the flow which came up during the discussions, a flow was finalised which would be used for implementation of the migration.

Task 5: Migration of the exists endpoint

After the successful completion of spike for the exists endpoint, I closed the spike and started a story ticket for the endpoint migration. For this, I implemented the changes in the code according to the flow we came out with during the spike ticket and the endpoint followed the whole feature development cycle which was discussed in the previous section.

Task 6: App Scan for basic get endpoint

This time I was required to provide the app scan document for the migration endpoint that I implemented in task 2. I prepared the app scan in the similar fashion as task 1 and submitted it.

Task 7: Demo for the basic by user id get endpoint

After everything from spike to story was completed for this endpoint it was supposed to be moved to production environment. Before that, we hold a demo of the work that has been going on in the team on a bi-weekly basis to keep everyone updated. This also allows for some modifications that might be beneficiary for the feature before it actually moves to production. I was required to give the demo of the API that I worked on in this endpoint to all the team members (onshore and offshore). It included presenting the work I did and the responses that I am getting through the API followed by questions by team members and managers if they had any.

Task 8: Spike for get by user id endpoint migration

The next migration which was required to be done was another get endpoint. This one gave out more details than the one I completed in task 2 and interacted with two services called EC to which the migration was to be done and CXP service, from which some part of the data was to be taken. During this spike, I was required to analyse the different endpoint that EC provides and extract as many fields in the contract as I could from it. Then I had to look up at the CXP endpoints and figure out an optimized way to extract the data required to fill the contract. During this discovery period, I connected with the EC team to propose a flow in which we could migrate the endpoint as it was to be migrated to their service. I also reached out to CXP service team to come up with solutions and finding a different way of calling their service which would help in reducing the times for the request. After everything was finalised and a proper flow was decided, I got the flow diagram approved and the spike was closed.

Task 9: Migration of get by user id endpoint

Similar to task 5, after completion of the spike I implemented the code changes that were required according to the solutions we came up with in the task 7. The feature was then moved to testing which will be tested by one of the testers in the team and I would be required to make changes in the implementation if any are necessary.

Other Tasks:

Apart from the major development and spike tasks that I continue working on, we keep doing some miscellaneous tasks like investigating certain bugs that we face or are reported by concerning teams. We keep communications and answer the questions raised by different teams which we are working with and work together with them to come up with solutions that their services might require. We also need to address any issues caused in any other services due to our possible actions and figure them out.

Apart from these, though there is a different team that handles performance and load testing, doing performance testing and analysing our API's is also one of the task that we perform in order to create API's that have low response time and high availability and avoid any cascading failures in the services.

Being online for on call support in the support rotations and handling alerts is also one of the tasks that we perform daily during rotations.

CHAPTER 6: CONCLUSION

The services that we develop and handle are a crucial part of the users shopping experience in whenever they come to the client's platform for the first time. Maintaining these services and ensuring that there is no downtime in the production websites as small downtimes can effect thousands to tens of thousands of users and their experience over the platform becomes an important task which the team tries to fulfil by providing high tolerance and well tested feature codes. Every aspect of any feature is carefully designed and implemented to work cohesively with all the other services ensuring a strong reliable service software system.

During this internship, I got to learn a lot of tools that are used by developers in order to attain different functionalities in a software lifecycle. Thanks to the bigger team that I got a chance to work with, which was very supportive and helped me through everything, I was able to develop my confidence and communication skills throughout the internship while providing optimized implementation of code. Thanks to ZopSmart for providing the opportunity to learn and grow while working with such great teams and keeping that trust in me to achieve what was required.

REFERENCES

- 1) Baeldung. 2022. Java, Spring and Web Development tutorials. [online] Available at: <<https://www.baeldung.com/>>
- 2) www.javatpoint.com. 2022. Tutorials - Javatpoint. [online] Available at: <<https://www.javatpoint.com/>>
- 3) Docker. 2022. Home - Docker. [online] Available at: <<https://www.docker.com/>>
- 4) GeeksforGeeks. 2022. GeeksforGeeks | A computer science portal for geeks. [online] Available at: <<https://www.geeksforgeeks.org/>> [Accessed 12 May 2022].
- 5) Stack Overflow. 2022. Stack Overflow - Where Developers Learn, Share, & Build Careers. [online] Available at: <<https://stackoverflow.com>>

INTERNSHIP CERTIFICATE

Date: 06/05/2022

TO WHOMSOEVER IT MAY CONCERN

This is to certify that Sanket Sharma has done his internship in software development at ZopSmart Technology Pvt Ltd., Bengaluru, from 03/01/2022 to 06/05/2022 for the duration of 4 months.

He has worked on a client project titled Digital Account Profiles. This project was aimed at managing CRUD on the client user profiles and finding and resolving any occurring bugs. As part of the project, he has worked in migrating endpoints, resolving bugs, endpoint cutovers and other duties related to the team.

During his internship he has demonstrated his skills with self-motivation to learn new skills. His performance exceeded our expectations, and he was able to complete the project on time.

We wish him all the best for his upcoming career.

ZopSmart Technology Pvt Ltd.

Rashmi S

Manager- Human Resources

