OYO Vacation Homes (Full Stack Development)

Parthkumar Patel
(201601024*)* DA-IICT
Gandhinagar, Gujarat, India
201601024@daiict.ac.in

Off Campus Mentor
Mr. Kamalesh Maurya
Oravel Stays Private Limited

On Campus Mentor
Prof. Jaideep Mulherkar
DA-IICT, Gandhinagar

***Abstract*— In my B.Tech. Project I have worked with OYO Rooms which is the world's third largest and fastest growing Indian hotel chain. Oravel Stays Private Limited, which is the company name under which OYO rooms operates on, has diversified its product portfolio with subsidiaries like Oyo Rooms, Oyo Life, Oyo Vacation Homes (which acquired Leisure Group in 2019) and Weddingz.in. OYO Vacation Homes team is responsible for managing the vacation homes, managed by Oyo in India and abroad. It has three European subsidiaries (which were previously owned by Leisure Group) namely, Belvilla, Danland, DanCenter and Traum-Ferienwohnungen (abbreviated as Traum FeWo). I was assigned to the Traum FeWo team with my mentor Mr. Kamalesh Maurya. I can classify my work in three main tasks/projects.**

1) **Since a large number of properties onboard it is extremely important to manage large contents of webpages efficiently. To solve this problem, I developed a CMS (Content Management System) to manage the contents of the various webpages.**
2) **As we know COVID-19 crisis has hit hard to the hospitality industry including OYO, we had to focus on the cost-cutting process. So my task was identifying unnecessary or more costly services running on the company stack and report them by going through the code modules.**
3) **Implementation of new features on the website of Traum-Ferienwohnungen.**
4) **Migrating the invoicing service from the Classic monolithic codebase into a separate microservice.**

***Keywords— Full stack Development,* CMS, Cost-Cutting, ReactJs, Springboot, SEO Page, Gantt chart, Incremental Waterfall Model, Monolithic, Microservice**

I. INTRODUCTION

I started my journey with OYO Rooms in the month of January 2020. There are various teams working at OYO which includes HMS (Hotel Management Service), Tech Support, DevOps, Online Reputation Management team, Payroll Team, HR Team, Search, Pricing, Growth, OTA, CEE Platform Tech, Supply Onboarding, CST (Call Centre Tech), Fintech Platform, OYO Life, OYO Vacation Homes, Payments, Website Team and Android team. I was part of the OYO Vacation Homes (Traum-Ferienwohnungen) team.
Team Culture at Traum-Ferienwohnungen:

- Daily Standup Meets / Daily Sync-up meets with the team members, mentor and manager to discuss the various issues faced in the project development.
- Frequent Knowledge Transfer sessions with the Traum FeWo development team based in Germany, to ensure rewriting the code becomes easier.
- Town halls with CEO and CTO for gaining knowledge about the latest developments in the company.

Projects/Tasks I have worked on are as follows:

A. User complaint registration and technician's task management system

This was a mock project to be completed in a week to get ourselves familiar with the technologies used and to train us for the fast-pacing startup culture. The system was developed to help the operating manager of the property to smoothly run the operations of the property by registering any issue with the property as soon as it occurs and get it fixed. The system provides a portal to the user to raise tickets regarding any category of issues with the property like appliance issues, furniture issues, utility issues, etc. There are also subcategories for each category of issues mentioned above like TV issues, AC issues, Washing Machine issues, etc. under the category of appliance issues. There is a specialist technician for each subcategory of issues. As soon as the user submits the issue a task is created. Each task created has one these states at any point in time – Assigned, InProgress and Completed. New Task is assigned to the technician who has the proficiency for that issue and has the least number of assigned and in progress tasks (combined) at that particular time instance. Also, System has a portal where the user can enter the technician id (Every technician has a unique id associated with her/him) and see all the tasks associated with that particular technician. The user can click on a particular task and see all the details associated with that task. Moreover, the user can update the state of the task from Assigned to InProgress and from InProgress to Completed according to the current state of the task. VueJs was used for the development of the front-end for this project. The backend was developed using PHP Laravel Framework and MySQL was used as a database for this project.

B. Content Management System (CMS)

The Main goal behind developing this project was to effectively manage the contents for different types of webpages. The system provides a frontend platform for entering the data for various types of pages for different regions/cities. This data will be stored in the backend database and we can use this data for rendering pages when needed. Also, the system provides functionalities of seeing these pages on the dashboard, searching for a particular page and edit the data of that particular page which was entered earlier.

C. Cost-cutting Process

This process is being done as a response of the prevailing COVID-19 crisis and saving the resource cost as much as we can to maintain the liquidity in the accounts of the company. Basic idea was to find out all the unused, partially used or more costly resources and report them. From the reported resources, some resources that are unused were removed and others, which were costing high, were downscaled or replaced with some cheaper resources.

D. Implementation of new features

Price Filter and Sorting Bar are the two new features that were added/modified for the website of Traum-Ferienwohnungen [1] by me. Both features are implemented for better user experience and user convenience. The Price Filter feature is especially implemented for price- conscious users whereas the Sorting Bar feature is modified so

that users can quickly sort the properties according to different attributes.

E. Migration of services

Currently company's technology stack has a monolithic architecture, which means there is a common codebase for every service running on the company stack. Monolithic architecture has some drawbacks and to tackle that we are migrating its services from monolithic architecture to micro service architecture, which means there will be a separate codebase for every service. I have worked on the migration of invoicing service.

II. CMS (DESIGN AND IMPLEMENTATION)

A. Problem Definition

Traum-Ferienwohnungen is an online vacation rental platform based out of Germany, which lists vacation rentals, apartments, villas, holiday homes, etc. from property owners and connects them with the guests who are seeking accommodation through its platform. It currently does not have a content management system. This site is in two main languages namely German and English. To make the management of the contents of both the website, we need a content management system. The main aim of the CMS currently is to allow adding of SEO pages. Currently, the conversion rate which means the number of hits on the website to the number of bookings is very low. In order to optimize it, we aim to implement SEO optimization. For this to happen new SEO pages for all the locations are to be added. This is currently done manually by directly adding data to the already existing PHP code (due to lack of CMS). Currently over 100 SEO pages have to be added, doing it manually is quite cumbersome. So a CMS will help in organizing the content of these SEO pages which are being added manually, later these content will be fetched from the database directly instead of injecting the data to the PHP code hence making this process automated and can be done by anyone. CMS will also ease the process of adding and managing data.

B. Objectives

The main objectives of a CMS include:

- It enables non-technically minded users (Business people, Home owners, etc.) to upload and modify content themselves, without having to understand programming languages such as HTML or PHP.
- Automate the process of adding SEO regions.
- To provide a better user interface to the people and reduce the possible third party cost.

C. Product Scope

Traum Fereignwohnugan is a website for booking vacation homes in Europe. From the technical perspective, this has two aspects. One is the website which the users see and book vacation homes from and the other is the home owner's portal where the vacation home/villa owners list their homes on the website. Both of these teams will require a CMS to drive content. So this will be used by both the teams (Home owners and the guest team) of Traum Fereignwohnugan. In the present website the content resides in one of the folders and inside a PHP file. So currently the Tech team has a lot of load when it comes to modifying the content. Every time a new requirement comes from the product team regarding changing a particular piece of content which may be regarding new offers/coupons which requires no technical knowledge and can be done by the product team. So this CMS will enable any authorized person to modify the contents without depending on the tech teams to do it.

D. Gantt Chart for the system

Figure 1 shows the Gantt chart for the development process of CMS, which basically depicts the timeline of the project.

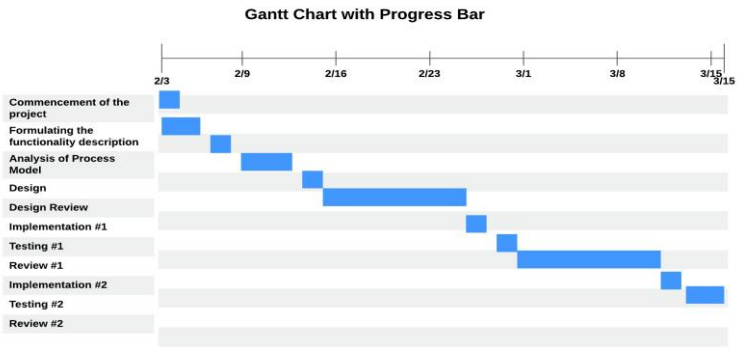

Fig. 1. Gantt Chart for CMS

E. Process Model of the system

The process model that I have selected for the CMS project is Incremental Waterfall. There exists a clear understanding of the requirements of the product. The architecture that is developed in the initial stages is stable and in a usable condition. The requirements are divided into different subsets and each release will contain the implementation of one of those subsets. Using this model will lead to a relatively lesser cost as I can start working on the next phase of the product as soon as the work of the current phase gets completed; a similar analogy to a pipeline. I shall start working on the succeeding phases based on the feedback received from the users and new requirements would be included in the future releases.

F. Design of the system

The whole website is divided into the following components.

1) Listing Page
   - Listing Page has all the web pages list which are created for different regions/cities.
   - It also has a search bar with autocomplete functionality using which we can search a page by its page name.
   - Listing has these columns: Name (Region/City Name), page type like SEO, landing, etc., template type like fixed, editable, etc., description and an edit button.
   - This page has an Add Page button for adding a new page.
   - When a user clicks on this button she/he will be prompted to select page type like SEO, Landing, etc. According to the selected page type user will be prompted to selected template type like fixed, editable, etc. For fixed template structure of the template is pre-defined and the user cannot edit the sections of the template but can only edit the contents of the template. For editable template the template structure is pre-defined but the user can add/remove/move sections as per requirement.
   - When the user clicks add button content creation page will be rendered according to the selected page type and template type.
2) Content Creation Page
   - Here user fills in all the details related to the page.
   - The page has different tabs for different languages so that the user can fill in the data in multiple languages.
   - There are some global fields like page name, URL of the created page and status of the page, which are independent of the language and rendered only once.
   - All other fields are rendered in all the language tabs according to the page type and template type.
3) Page Content Updation
   - When the user clicks on a page on the listing page, the already filled data will be loaded and the user can edit it.

G. Implementation of the system

1) API first approach

The approach that was used to design and implement this website was API first approach. Under this approach, API development is considered to be the primary step. APIs expose the different endpoints that are required for creating, retrieving, updating and deleting information on the server hosted. Thus different get, post and put (update) APIs were developed using JAVA Springboot Framework for different tasks like adding a new page, updating a page for the given page name, listing down all the pages created, getting all the page related details for the given page name, getting the page data according to the language for the given page name, searching a page according to the page name, creating a template, updating the template, fetching template structure for given page type and template type, getting different template types for the given page type, fetching all the supported page types, template types and languages, etc. In the first step, these APIs were built using the mock server system provided by Postman App and then the actual servers were implemented. Example API call for adding a page:

POST: http://localhost:3002/pages

```
{
        "name" : "Lisbon",
        "pageType": "landing",
        "templateType" : "editable",
        "description" : "Lisbon is the capital of Portugal",
        "status" : "active",
        "globalFields" : [
                {
                        "url" : "https://sample.com"
                }
                ],
        "listOfPageData" : [{
                "language" : "English",
                "pageData" : {
                        "key" : "value"
                }}
                ]
}
```

2) Website implementation (Front–end)

Front-end means the UI and UX elements that the user faces. It includes all the HTML elements such as buttons, tables, links, etc. that is visible to the user on the browser. Figure 2 shows the listing page of the website. It has a list of all the created pages, a search bar and an Add page button. When this page is rendered an API call is made for listing down all the pages created.

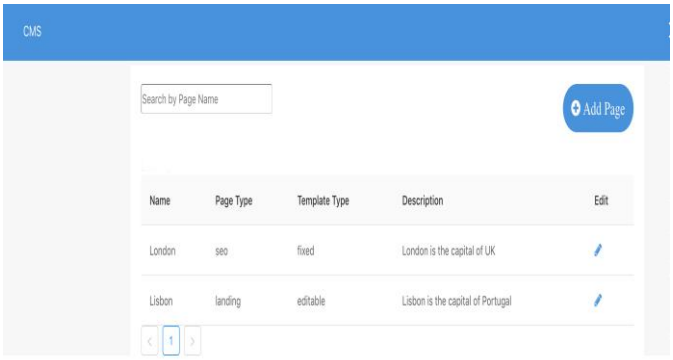

Fig. 2. Listing Page of the website

When the user clicks on Add Page button a dialog box will be opened as shown in Figure 3 and an API is called for fetching all the page types.

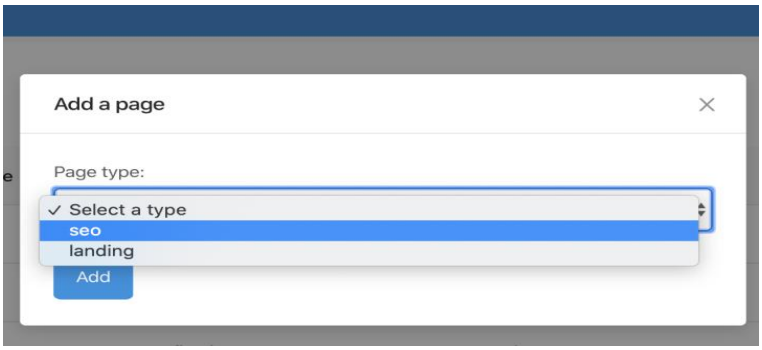

Fig. 3. Dialog Box showing different page types

When the user selects any of the page types an API call is made and all the template types, which are available for that page type will be shown in the dialog box as shown in Figure 4.

Fig. 4. Different template types according to the chosen page type

When the user clicks on the Add button an API call for fetching initial template structure according to the given page type and template type is made and Content Creation Page is rendered accordingly. For example Figure 5 shows Content Creation Page for SEO page type and editable template type.

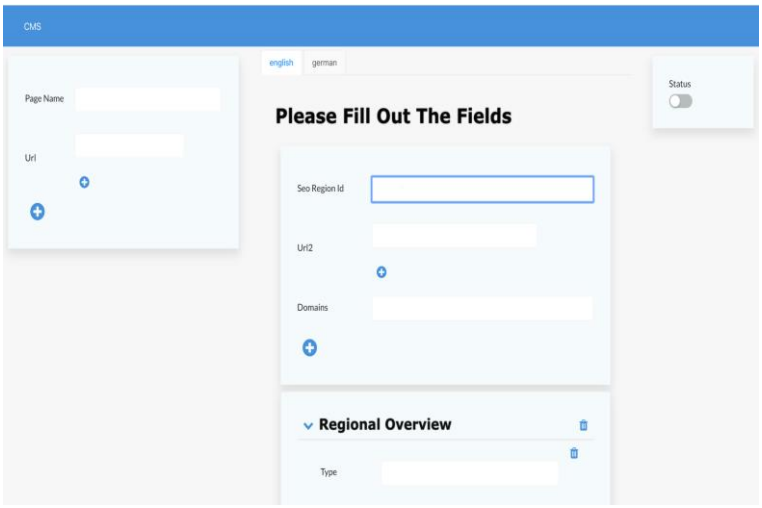

Fig. 5. (a) Content Creation Page Part – I

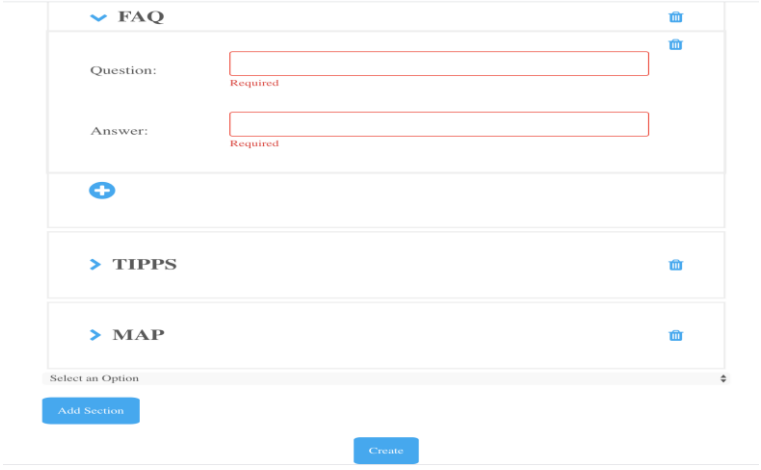

Fig. 5. (b) Content Creation Page Part – II

When the user clicks on the Create button, which is shown in Figure 5 (b), a POST API is called and page data will be saved in the database. The user can also update the data of a particular page by clicking on the pencil icon associated with it on the listing page as shown in Figure 2. For this, a GET API will be called for loading the data and a PUT API will be called for updating the data.

3) Website implementation (Back–end)

The Back-end is the business logic required for the website to run. It includes making API calls to the server, building APIs, serve HTML files, performing logical operations on the data received, database management, defining different routes for different pages of the website, etc. Backend APIs for CMS are already mentioned in the API first approach section. When we call these APIs from the frontend, data will be fetched, stored or updated according to the API called (GET, POST or PUT) by interacting with the database. We can introduce a BFF (Backend For Frontend) for interaction between frontend and backend. This aspect will be covered in the future implementations of CMS.

In CMS Back-end there are three models (tables) defined as follows:

- Template Model – It has the basic skeleton of the template. It has pageType, templateType, globalFields and templateStructure as fields. templateType is defined as an ENUM containing fixed and editable. 'templateStructure' is a JSON object which contains all the section labels which are to be rendered on the Page Content Page. 'pageType' and 'templateType' together uniquely define a record of this model.
- Page Model – It has pageName, pageType, templateType, description, status, globalFields and listOfPageData as fields where listofPageData has the list of page data according to different language. 'pageName' uniquely define an entity of this model.
- TemplatePage Model – It has pageType, templateType and language as fields where templateType and language are ENUMs. As discussed earlier templateType contains fixed and editable as the values of ENUM. 'language' has English and German as ENUM values.

H. Tech Stack of the system

Table I shows various technologies used in the project.

Component	Technologies
Front-end	ReactJs, Redux, Ant- Design, Axios, Formik, Yup
Back-end	JAVA SpringBoot, Postman, Junit
Database	MongoDB

Table I. Different technologies used for the implementation of CMS

- Front-end: ReactJs [2] is a javascript library used for UI development. All the webpages of CMS are designed using ReactJS and their rendering logic is also defined using ReactJs. Redux is used to manage the state of the application. For getting different type of stylish components for UI Ant-Design is used. Axios is used for the API call to a particular URL by front-end. Formik and Yup are used in the frontend for handling form data and validation respectively.
- Back-end: The CMS Backend is developed using JAVA SpringBoot [3] Framework. According to the Framework norms, Models for every table in the database are developed. For API declaration controllers for each model are developed. Methods for APIs are defined in different service files according to model entities. For CRUD Operations with the database repository files are created. Postman is used for testing the APIs before integration to front-end by mocking the data. Unit Test cases for different APIs are written using Junit.
- Database: MongoDB is used as the database for the development of CMS. MongoDB is a document- based NoSQL database, which stores data as JSON. Page, Template and TemplatePage are the tables, which are stored in the database for CMS.

III. COST REVIEW AND COST-CUTTING PROCESS

As mentioned earlier there was a dire need for cost-saving due to the current financial crisis created by COVID-19. The goal was identifying resources, which are costlier but are not used to its full potential and can be downscaled, replaced with other resources or deleted. My task was identifying those resources by going through all the system and project code and report them as well as to help the team to carry out the cost reduction process. The Process was carried out across platform and IT projects. Moreover, the front-end website of Traum-Ferienwohnungen is also optimized in order to reduce the cost.

Table II shows the cost reduction per month for some of the resources of platform projects.

Action Item	Estimated Cost Reduction/Month
Kameleoon A/B testing	€ 1780
Scaling Down Database + HA disable - Prod	€ 1605
Change Staging pods to pre-emptive (OnSpot)	€ 1500
Check all the Disk which are not shown in use 3620 -- SSD (600 euro per month) 10580 -- Standard Disk (400 euro per month)	€ 1000
Change ES clusters switch to single node clusters	€ 470
Downscale Core Data ElasticSearch	€ 450
Remove HA from Redis	€ 420
Delete Extra Databases (staging-hms-tmp)	€ 400
Scheduled downtime on staging for 12hrs. (6 pm - 4.30 am UTC)	€ 200
Decrease Sphinx nodes to decrease memory usage	€ 190
Merge all the staging dbs into single instance	€ 150
Highway: Scale down to save memory	€ 30
Highway-API add autoscaling to reduce wasted CPU requests	€ 30

Table II. Cost Reduction per month for some of the resources of platform projects

The cost cutting process results for IT projects is shown in Table III as below.

Action Item	Estimated Cost Reduction/Month
Changing MS Office 365 E3 to Microsoft 365 Business for 125 accounts	€ 350
Changing MS Office 365 E3 to Microsoft 365 E1 for 35 accounts	€ 455
Terminating the license for Gliffy - a Confluence plug-in	€ 366

Table III. Cost Reduction per month for IT Projects

As mentioned in Table III we switched to cheaper versions of MS Office and terminated licenses for some soft wares that are unused at present for saving costs. The cost-saving process for the front–end website of Traum-Ferienwohnungen was done by doing below tasks:

A. *Removing maps from SEO pages for cost-saving*

Currently, there are 5 million hits for the maps on the website which has increased the cost because all these maps are currently rendered using Mapbox and leaflet and there are many places where maps are being used. So maps from SEO pages are removed and their impact is to be evaluated. A click to open feature was implemented for all the other maps on the website, in which maps don't get loaded on every page load. All these button events are pushed to Tealium and tracked on Google analytics. This feature is currently live.

B. *Changing scalar tiles to vector tiles in maps in order to reduce the cost*

Traditionally, maps are created from image tiles. Currently maps in the website use scalar tiles which means the maps are viewed as .png format images. But the number of requests is huge which leads to a large number of API requests and hence the cost. So moving to the vector tiles considerably reduces the number of requests. One of the main advantages with vector tiles in terms of cost is that there is no need for zoom levels when users zoom and pan throughout all scales. The vector tiles have the .pbf format. I have implemented this using mapbox GL JS [4], which has different pricing compared to the current mapbox pricing for raster tiles. Figure 6 and 7 shows the number of API calls when we use vector tiles and scalar/raster tiles respectively.

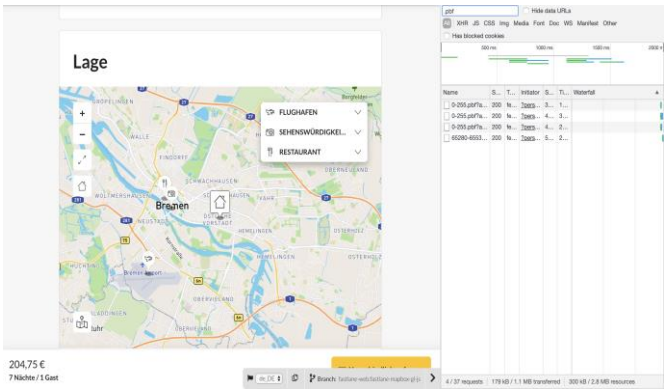

Fig. 6. Using vector tiles there are 4 API calls

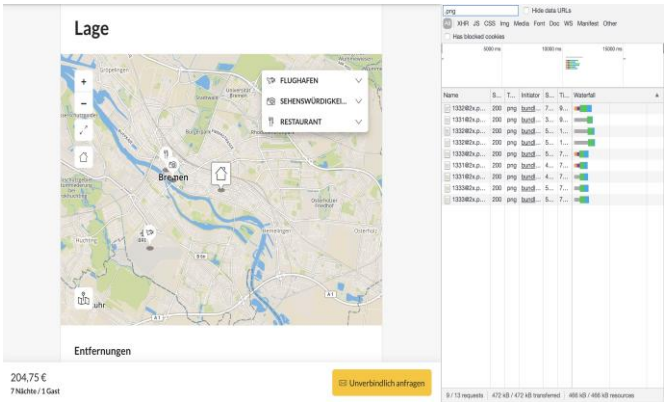

Fig. 7. Using raster tiles there are 9 API calls

IV. FEATURE IMPLEMENTATION

A. *Price Filter*

Currently, we do not have a sort by price option on the website. Sorting by price is an important demand by our users in order to find a holiday home that matches their budget. The expectation is an increase in conversion due to this change. I have contributed to the frontend part of it, which involved coding a range slider which helps the user to choose a range and these values get passed as the query parameters. These changes were added to mobile, tablet and desktop view of the site. Figure 8 and 9 shows the implemented price slider for desktop view and mobile/tablet view respectively.

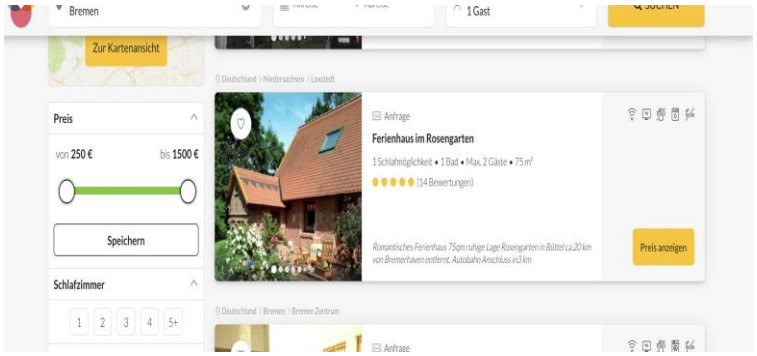

Fig. 8. Price Slider for Desktop View

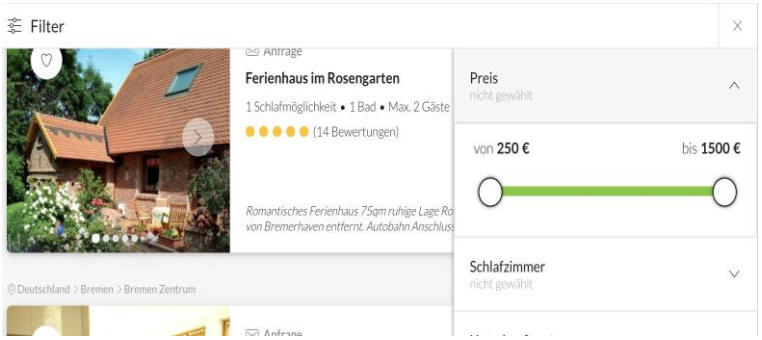

Fig. 9. Price Slider for Mobile/Tablet View

Price-conscious web users may choose to list the products in order of price, from cheapest to most expensive. As consumers, we have been trained; at least to some degree that price correlates closely with quality and most expensive apartment promises good ambiance. So this feature is particularly important.

B. *Sorting Bar*

The new sorting bar was to be implemented for desktop devices. The old and new sorting bars are shown in Figure 10 and 11 respectively.

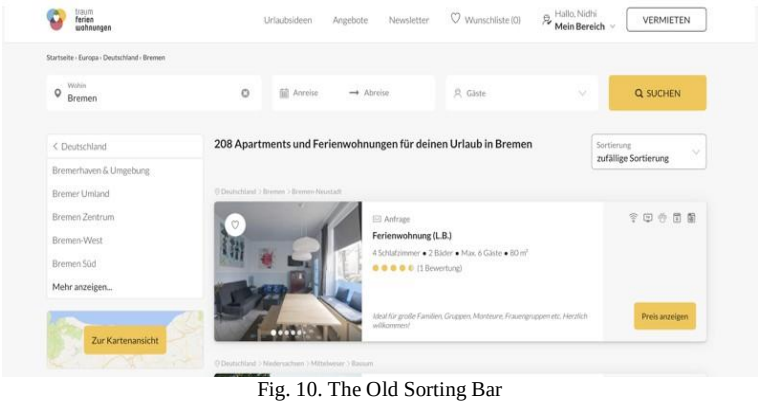

Fig. 10. The Old Sorting Bar

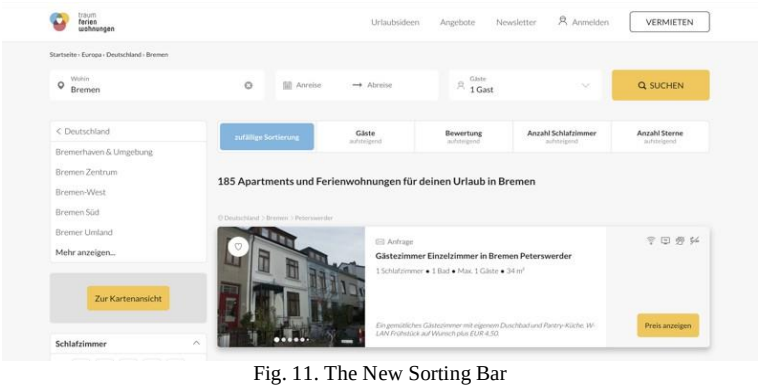

Fig. 11. The New Sorting Bar

The new sorting bar won't be a classical select where two clicks are needed, one to open the select and the other to select the actual sorting option. Instead, the aim is to display a new sorting bar where

all options are already visible for the user. One click and the SRL (Search Result of Listings) sorting changes. To test this AB test is implemented. Currently, AB test is running on Kameleon [5]. Through the AB test we are tracking on every sorting option to see what's important for the user. Current options are sort price, sort bedrooms, sort stars, sort ratings and sort guests.

V. MIGRATION OF THE INVOICING SERVICE

As discussed earlier, monolithic architecture has some drawbacks like it cannot adapt to new technologies easily because the change will affect the whole application and that will become cost and time consuming. Also, Understanding of the code will become extremely difficult when application scales up. These drawbacks are taken care of in micro service architecture because of its separate codebase for every service.

My task was to migrate the invoicing service from the legacy code into a standalone micro service. The original code was written in PHP on a custom framework and the new micro service is written in Java on Spring Boot Framework.

Learning Outcomes:

- Creating Spring Boot Java applications
- Database Access Object concepts
- Creating APIs to act as a bridge for the flow of data between the legacy code and micro service
- Schedulers to schedule certain micro service functions

Softwares Learned:

MySQL: Relational Database Management System tool used to create the database for invoicing service
Java and Spring Boot: Spring Boot framework was used to develop the service. Salient features include runtime dependency injection and modularity
Quartz Scheduler Tool: Quartz Tool APIs were used to schedule certain invoicing services tasks. Quartz tool is similar to cronjob service but is much more robust and functional.

VI. FUTURE WORK

Some of the future improvements/functionalities that can be added are as below:

- For CMS, A template structure has to be defined for every kind of page.
- For making pages searchable by certain more fields other than page name backend API is ready but needs to implement the Front-end part in the CMS.
- For the development of the CMS, a standard three-tier architecture is used but in the next addition, a BFF (Backend For Frontend) will be introduced between the backend and the frontend.
- UI improvements for CMS.
- For cost-saving process more resources where cost can be saved needs to be identified.
- Aggregation to get the minimum and maximum value for the price filter needs to be implemented from the backend.
- Migration of more services from Monolithic architecture to Microservice architecture

VII. CONCLUSION

The industrial internship at OYO Rooms was a great learning and enthralling experience for me. Things that I learned during my internship period are as follows:

- Worked in a very fast expanding startup, which enabled me to get exposure of fast pacing startup culture. I got to learn the complete end-to-end product development from very scratch which includes gathering requirements for the product, deciding the development model of the project, actual coding, testing, deployment of the project on the production and finally launching the product (Complete Software Development Life Cycle).
- Got to learn how to write an efficient and production level code by following the best industry practices.
- The main learning experience was to be a part of the cost-saving team. Got understanding of every project and technology which is running on OYO stack.
- Due to the implementation of CMS front-end got to master one of the trending front-end development libraries called ReactJS about which I had a little knowledge before the internship.
- Got the opportunity to develop the CMS backend from the scratch using JAVA SpringBoot Framework.
- I wrote test cases for the CMS project using Junit, which enabled me to understand how testing helps to maintain the sanity of the code.
- Learned how to interact in the team because of working with the international team based in Germany.
- At OYO, there is a practice of presenting your product to the senior managers and directors after completion of the development. This helped me to improve my presentation skills.

ACKNOWLEDGEMENT

I would like to thank my mentor, Mr. Kamalesh Maurya, for his constant support and guidance throughout my internship journey. Whenever I was stuck on something he was always there to help me with my daily struggles. I am very grateful to my Engineering Manager Mr. Ritesh Thounaojam for giving me an opportunity to work with the OYO Vacation Homes (Tech.) team. His advice on project management and organizational behavior helped me a lot during my internship. I would like to thank my On Campus mentor Prof. Jaideep Mulherkar for giving me the chance to gain experience like a working professional and also for his constant supervision. Lastly, I would also like to thank my all colleagues for their support.

REFERANCES

[1] Traum Ferienwohnungen website
https://www.traum-ferienwohnungen.de/

[2] ReactJs Tutorial
https://reactjs.org/tutorial/tutorial.html

[3] SpringBoot Tutorial
https://spring.io/guides/gs/spring-boot/

[4] Mapbox GL JS
https://docs.mapbox.com/mapbox-gl-js/api/

[5] Kameleoon website
https://www.kameleoon.com/en/platform/ab-testing