

Department of Computer Science & Engineering
Indian Institute of Information Technology Senapati

Venue Management System using QLO

(Course Code: CS 200)

PIYUSH KUMAR

18010107

Supervised by

DR. NAVANATH SAHARIA

24 June 2020

Abstract

QloApps is an open-source hotel reservation system and booking engine. With the help of QloApps, anyone can launch his hotel booking website in few minutes. This features sounds a bit related to venue management system. The main aim of my module is to provide an easy and simple interface to the customers to manage their venues under a click. Best hotel reservation system should have leading-edge technology and provide the basic functionality to hoteliers to increase their direct online bookings. In today's modern world where all are tech-savvy, responsive hotel reservation system can be accessed from anywhere at any time to book rooms and hotels. Similarly it will be great if we try to take the venue management services in the same flow or way.

Qlo is basically built upon MVC, also known as Model-View-Controller. It is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

Chapter 1

Introduction

This document is a report for the individual project "Venue Management System using Qlo". In this project I have created a custom module for Qlo to facilitate Venue Management and Reservation System. There are lots of venue management and reservation systems out there based on the web interface. But all those systems are not up to the mark. So this module is created by considering all the features that are required for a basic venue management system to successfully run. This module can be very handy to small business scale organisations. This module can be installed on any existing Qlo site.

In chapter one, I have discussed the basic introduction to the project. What is Qlo? What is this module used for? It will give a brief idea to the whole project.

In chapter two, I have discussed the existing venue management systems based on web. What different features and functionalities they provide to us. What features they are lacking?

In chapter three, I have discussed the system design for our Qlo module. How I started working? Which technologies have I used?

In chapter four, I have discussed the implementation of our module. How different functions are implemented. Which type of custom functions I have created? Which core functions of Qlo has been used?

The whole Qlo is based on PHP 7, therefore the venue management system module for Qlo will be developed using PHP 7. Qlo alone is just a hotel reservation system. There is no any functionalities for venue management or venue booking. So this is a great idea to create a simple module that will extend the current functionalities of Qlo by adding up venue management functionalities. In this module I will create different controllers using different hooks of Qlo. The two main types of controller I will be focussing on is front and admin controller. These two controllers will be the base of my module that will be serving different functions or functionalities. The most important thing

to note here is that, I had to work according to the MVC architecture and Symphony framework. As these are basic building blocks on which the Qlo is built up.

1.1 Problem statement

I had to develop a custom module for QLO to facilitate venue management. Venue management itself encloses venue reservation or booking along with management. A module is a very handy way to extend the functionality of any existing system. We can write our custom module to implement any functionality we want in that system. This is a really a great opportunity for the developers to tweak the system according to their need or build a module to help peoples out there waiting for new features.

1.2 Motivation behind selection of the project

From the begining of my B.Tech, I was attached a lot to web development. There is a special bond between me and web development. This project is completely based on web development, and web development is my the most favorite field to explore. In this project there is a lot of things to learn about web development, database, object-oriented programming in php, MVC framework e.t.c. So all these new things and I can say perks motivated me to select this project.

1.3 QLO Module

A Qlo module is a collection of files containing some functionality and is written in PHP. Because the module code executes within the context of the site, it can use all the functions and access all variables and structures of Qlo core. In fact, a module is no different from a regular PHP file that can be independently created and tested and then used to drive multiple functionalities.

This approach allows Qlo core to call at specific places certain functions de- fined in modules and enhance the functionality of core. The places where code can be executed are called "hooks" and are defined by a fixed interface. The basic functionality needed to develop a website is provided by Qlo in its core itself. The user just needs to enable it from the extension. The core modules make it easy to add content, publish them and create pages. The module gives the user full control over how they want their website's functionality.

All the additional functions user need in his site are provided by Qlo in the form of modules. Modules are the elements which provide QLo with its flexibility and make it one of the best CMS out there. The user can turn the features and functionality on by installing the module and can turn it off by uninstalling the module but before uninstalling, the user may need to remove data and configuration related to the feature or functionality. Each module

that is installed adds to the time needed to generate pages on your site, so it is a good idea to uninstall modules that are not needed.

1.4 Roadmap

I started working on this module development in a very organised fashion. In the beginning it was very uncommon and new to me. Slowly I started looking up the codebase of Qlo to get a better understanding of how this system works. As there are no documentation for Qlo, it was quite difficult in the beginning to get comfortable with the environment and the codebase.

After getting familiarized with Qlo. I started working on the basic functionalities of the module, like installation of the module, uninstallation of the module. Then I started triggering different functions on such events like install or uninstall.

After doing all these things I worked on the database to create a robust and perfect ER diagram to tackle every situation. After working on the ER diagram, I implemented that ER diagram on the Qlo module. Whenever the module is installed or uninstalled I trigger the respective functions to update or change the database.

After implementing the database part I started working on the different controllers and classes to provide all the basic features and functionalities like adding a venue, adding a feature to the venue, removing the venue, removing the features of the venue, updating a specific venue, updating a specific venue feature and a lot of stuffs.

After working on all these things I started working on the booking system, validating the checkin and checkout dates in the real time, creating different front views or user interface.

1.5 Your contribution

This complete module is created by me under the assistance of my supervisor, Dr. Navanath Saharia.

1.6 System/Software used

As I am developing the module for Qlo so its obvious thing that I have used Qlo. Qlo is written in PHP. So I have also used PHP as my main language to create this venue management system module.

Qlo is built upon MVC architecture, as I have discussed earlier that MVC is an architectural pattern to build to build robust and complex applications. So I have followed the MVC pattern throughout this development.

Qlo is built using Symfony framework. Symfony is a PHP web application framework and a set of reusable PHP components/libraries. It was published as free software on October 18, 2005 and released under the MIT license. So I have followed the symfony framework pattern throughout the development.

For the case of database, I have used mysql as its a very popular database software and also its a lightweight. Just being the popular doesn't let me choose this software, instead it fits perfect for this project that's why I choosed this software.

For building the User Interface I have used Smarty template files as it is the de-facto standard for building User Interface in symfony frsmework. Smarty is a web template system written in PHP. Smarty is primarily promoted as a tool for separation of concerns.[2] Smarty is intended to simplify compartmentalization, allowing the front-end of a web page to change separately from its back-end. Ideally, this lowers costs and minimizes the efforts associated with software maintenance. Smarty generates web content through the placement of special Smarty tags within a document. These tags are processed and substituted with other code. Tags are directives for Smarty that are enclosed by template delimiters. These directives can be variables, denoted by a dollar sign (\$), functions, logical or loop statements. Smarty allows PHP programmers to define custom functions that can be accessed using Smarty tags.

1.7 Implementation plan

Here is gantt chart which states how I have worked throughout these months. How and when I have implemented different features and functionalities. How much time did I devoted to this project. This gantt char gives a brief overview of all these things.

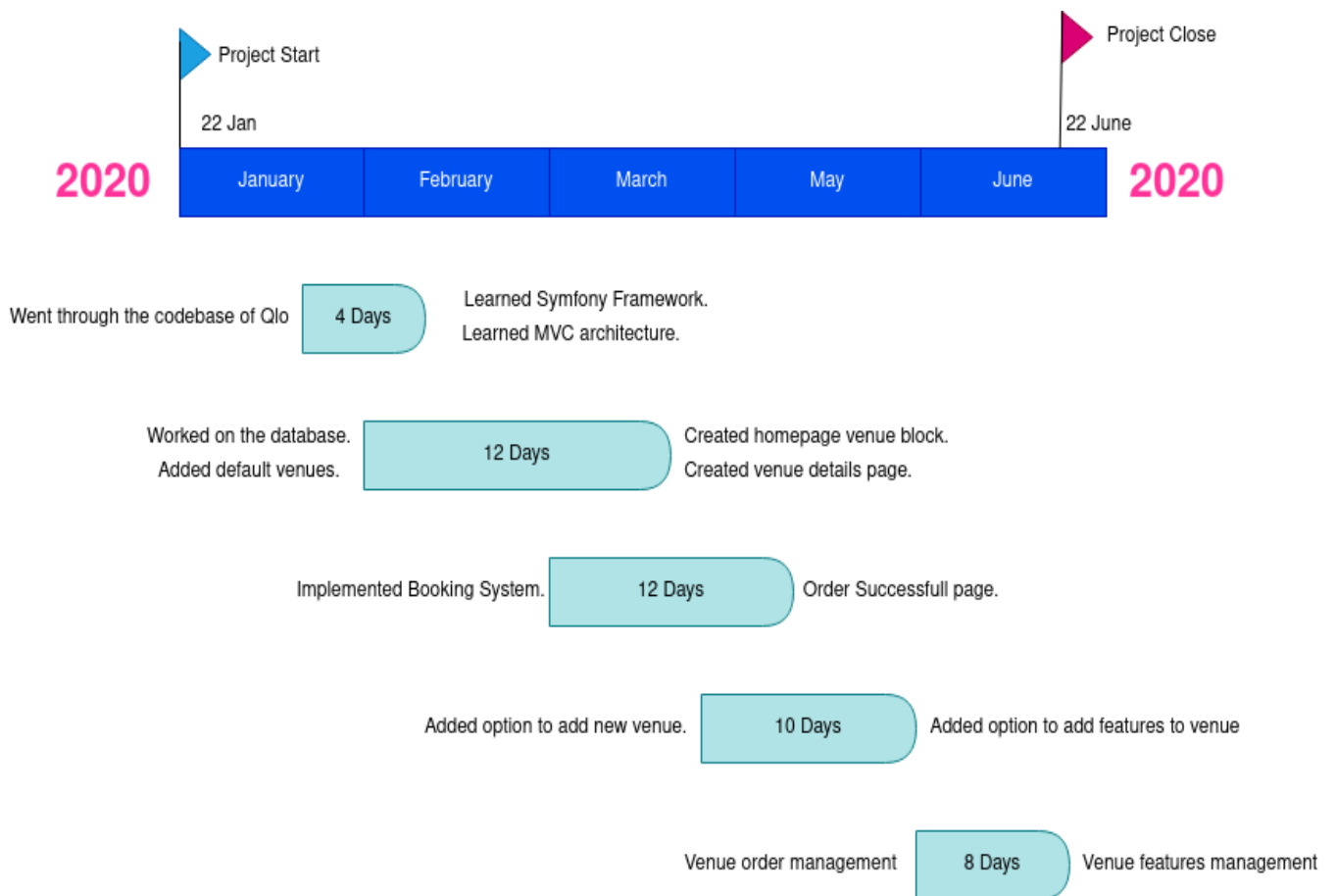


Figure 1.1: Gantt Chart

Chapter 2

Related work

2.1 Existing system study

There aren't a lot of venue management system based on web. Somehow I was able to find some of the popular venue management system that are based on web interface listed below-

- MeetingPackage
- Priava
- Planning Pod

All these 3 sites are based on web interface. Out of these 3 sites all are SMI business sites.

2.2 Features/functions of the studied systems.

Some of the prominent features of these websites are-

- Booking Management.
- Catering Management.
- Event Management.
- Reporting and analytics.
- User Management.
- Reservations Management.
- Search Venue.
- Free trial for a limited time.

2.3 Method(s) used for system study

1. Observation : The most reliable data are obtained through observations. So it's my first motive to analyze the in and out of these websites through various observations. Information received through interviews requires verification. People may not lie deliberately, but very often they misrepresent facts. So according to my point of view 'observation' is the best I can go for.

2.4 Comparative analysis

After a long analyzation of different features and aspects I came to find that all these features are average. They are implementing different functions in a very unmanaged fashion. They haven't also considered various security risks that can lead to their corrupt transactions. Some of the venue management system don't have the feature to book the venue for very future dates. Some venue management system don't give a reference to their customer for their bookings.

2.5 Summary

So with all the analysis and system study, I came to a result that most of the venue management systems doesn't meet the average requirements of the customers or users. They only provide the basic features to their customer. But as the technology grows or develop, there comes a lot of features and vulnerabilities that we as a developer need to consider and think about it seriously. We need to update our systems along with the development of the technology.

After finding out all the different features or functionalities that are lacking in the existing venue management system, I included all those lacking features in my venue management system. For the current scenario these features are enough to run a venue management system successfully.

Chapter 3

System Design

3.1 System design

Our system will have basically 2 main entry points. The first one is front view or front site from which users will be facilitated with different functionalities of the website, such as searching different venues, getting various details of different venues, making their booking of venues, etc. The second entry point is the admin controller or admin site from which only authorised users such as site admins will be allowed to enter. From this point the admin will be able to add different venues, manage their existing venues, activating or deactivating their venues, analyzing their analytics, etc.

3.1.1 Architecture

I have created a robust ER diagram for my module. There are basically 3 main or strong entity- venue, vfeatures, and booking. There are 2 1:N relationships between venue entity and vfeatures and booking. The ER diagram can be referred as shown in figure 3.1.

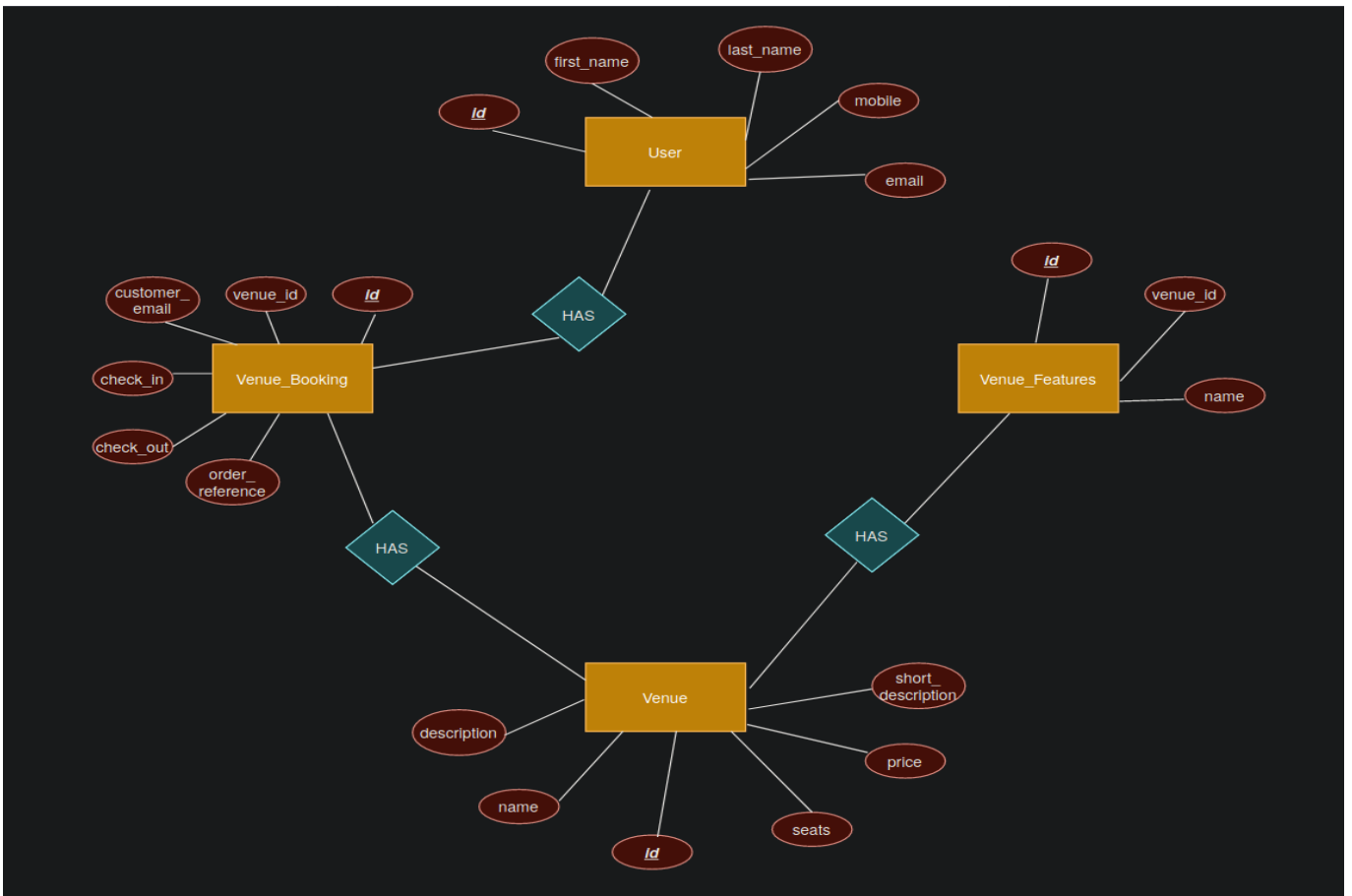


Figure 3.1: ER Diagram for Venue Management System

Chapter 4

Implementation

As I am developing the module for Qlo, which is completely written in PHP. Therefore I will be implementing my complete module using PHP. For creation of front view I will be creating separate templates file with .tpl extension. The main benefit of .tpl files are that they are highly customizable means we can write html codes along with php in a .tpl file. For the case of database I am using mysql, as it is very easy to use and also its a lightweight software.

4.1 Experimental set-up description

I have created a class diagram to better describe the venue management system module. The class diagram can be referred as shown in figure 4.1.

4.1.1 Function/Module/Class description

A clean and modular project is worth descriptive than a unmodular project. My whole project is itself a module for QLO. But still I have made it more modular by splitting different functions and classes into separate files as different modules. I have worked basically on two viewpoints-

- Front Panel
- Admin Panel

All the stuffs related to front view are categorised under front controller and all the stuffs related to admin view are categorised under admin controller. Qlo itself is based on MVC Framework which makes it more easier to arrange your files systematically making all your components very modular and easy to categorise. The main motive and functionality of these admin controllers are to provide easy management of different venues to the client. The 2nd admin controller will provide the functionality of managing different orders placed by customers in a very clean fashion.

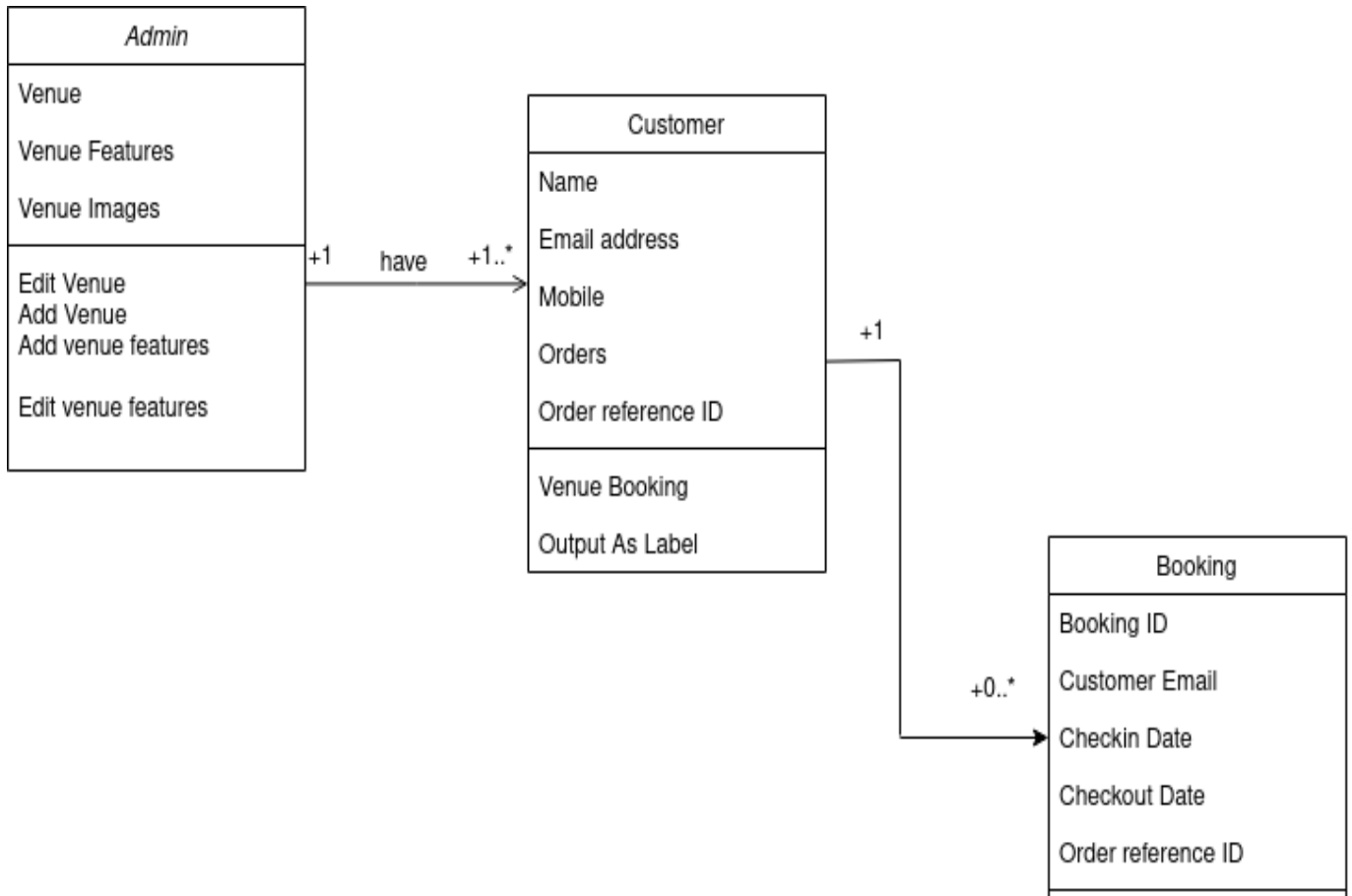


Figure 4.1: Edit existing venue

4.2 File Structure

4.2.1 classes

In the classes folder there are different classes to provide all the features and functionalities to the users. The main work of class is that it is triggered by the controllers when the user needs to do an action. The respective controller triggers the respective class to provide the function.

4.2.2 controllers

In the controllers folder there are different front and admin controllers to listen for any action or changes that happen on either the front-office or back-office. These controllers when gets triggered trigger its repective classes to provide that feature user needs.

4.2.3 sql

The sql folder contains all the database files for our module. There are two main files in this folder. First one is install.sql and the second one is the uninstall.sql. These files manage the database like setting up all the different

tables. The `install.sql` file will be called whenever the module gets installed to setup all the different tables in the database, similarly the `uninstall.sql` file will be called whenever the module gets uninstalled so as to remove all the existing tables related to our module.

4.2.4 views

In the `views` folder, all the templates files are stored to render different front views. In this template files I have written code in Javascript and PHP to achieve all the access to make the functionalities easier to implement.

4.2.5 alenia.php

This is the main file for our module. This is the entrypoint for which Qlo looks up to do further actions. This file holds all the reference to our other files to provide all different functions user needs.

4.2.6 checkout.php

This `checkout.php` file is a custom file in which I have implemented some needful functions for the checkout purpose.

4.3 Obtained result

Here are the final output images for the venue reservation system module.

Our Venues

Tofu helvetica leggings tattooed. Skateboard blue bottle green juice, brooklyn cardigan kitsch fap narwhal organic flexitarian.

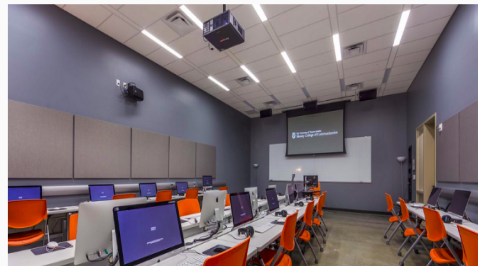


Auditorium

₹3500 / day

The bedding was hardly able to cover it and seemed ready to slide off any moment.

[Book Now](#)

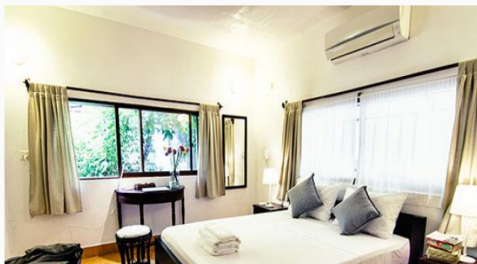


Computer Lab

₹2500 / day

The bedding was hardly able to cover it and seemed ready to slide off any moment.

[Book Now](#)



Guest House

₹4000 / day

The bedding was hardly able to cover it and seemed ready to slide off any moment.

[Book Now](#)



HB1

₹2500 / day

HB1 Boys Hostel HB1 Boys Hostel HB1 Boys Hostel HB1 Boys Hostel HB1 Boys Hostel

[Book Now](#)

Figure 4.2: Homepage Block

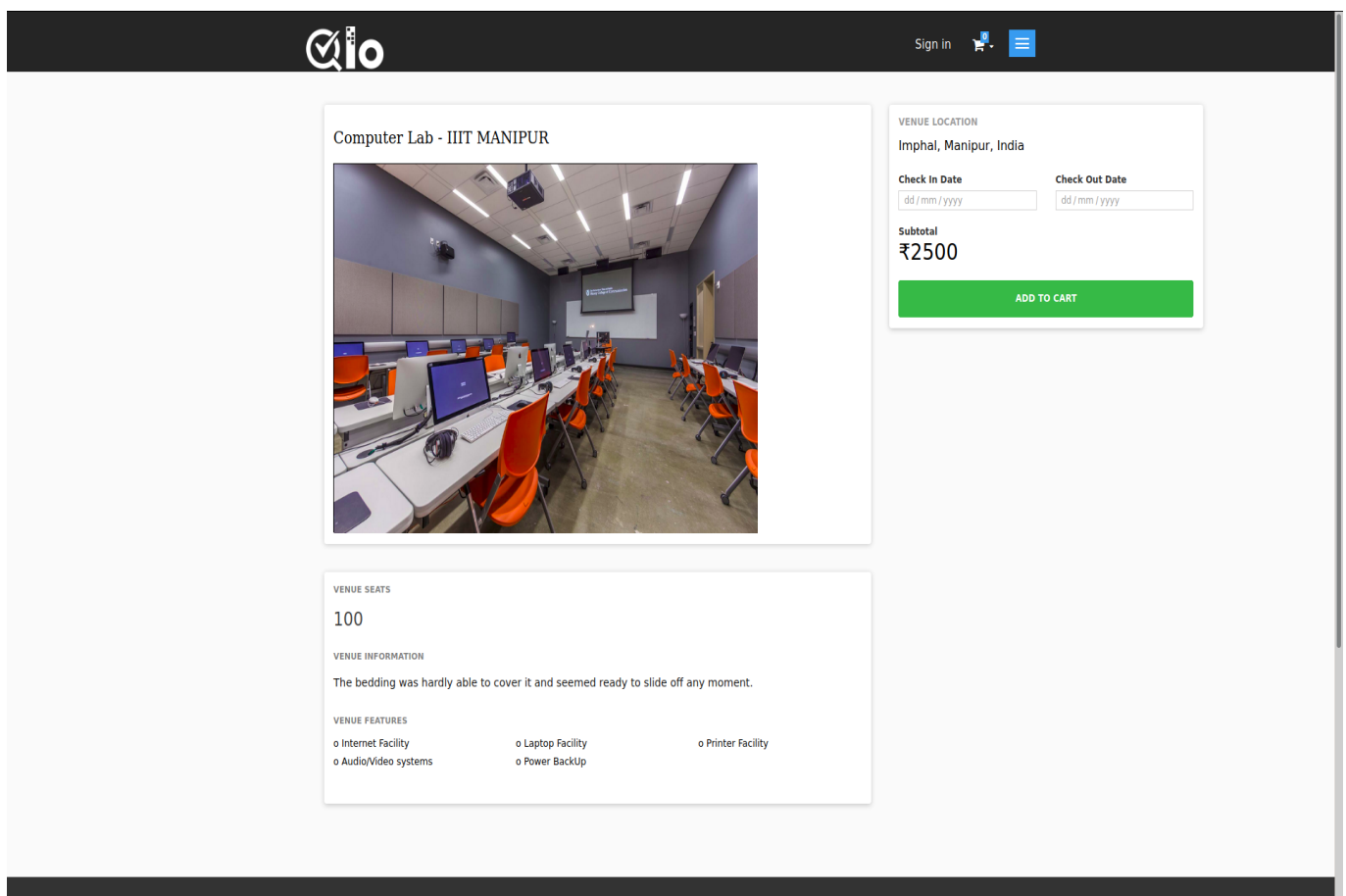


Figure 4.3: Venue details page

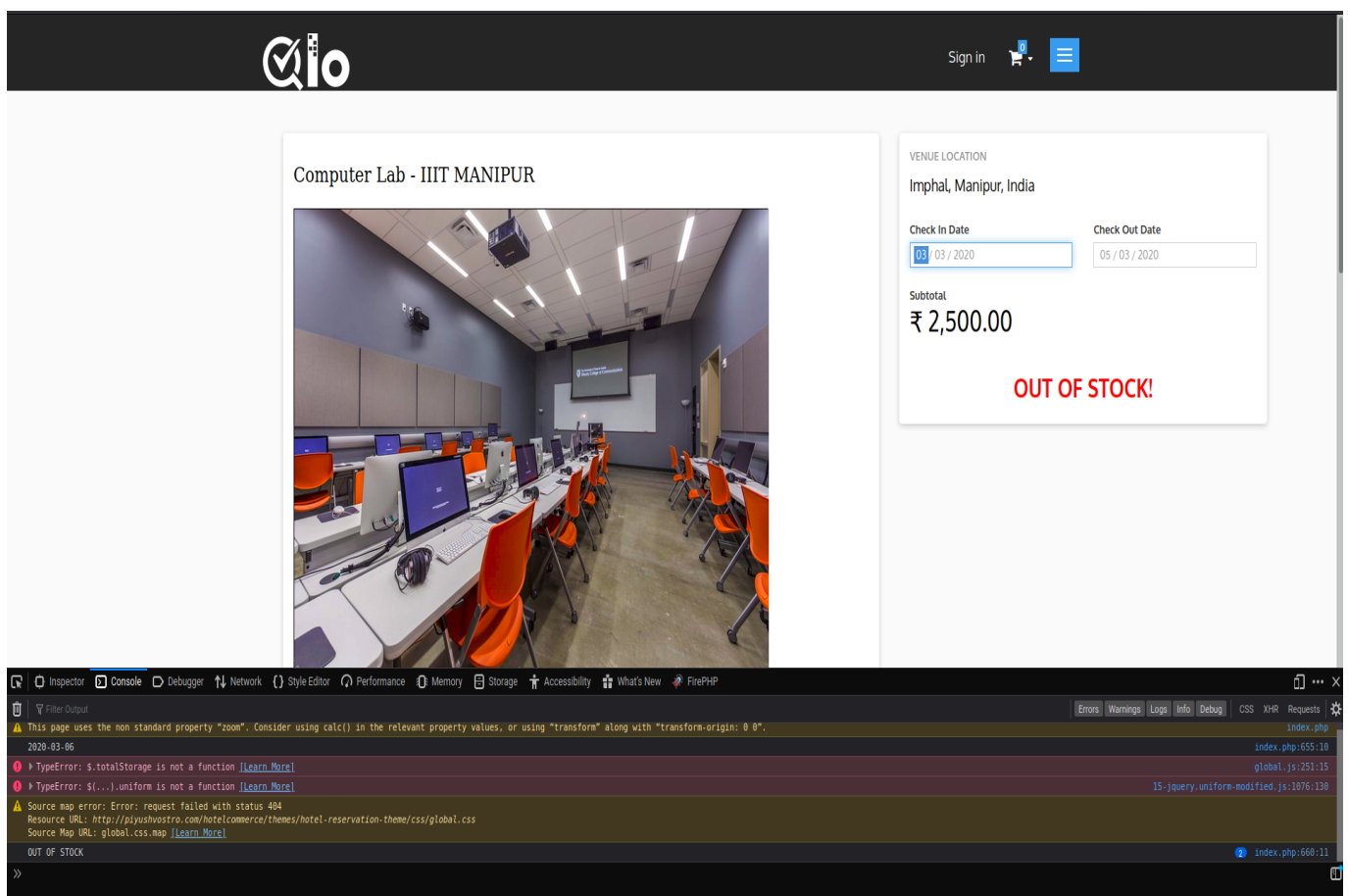



Figure 4.4: Out of stock error

Venue & Price Summary



Computer Lab

✓Internet Facility

✓Laptop Facility

✓Printer Facility

✓Audio/Video systems

✓Power BackUp

₹ 2500

2020-03-03 - 2020-03-04

Total Venue Price
(Incl.) all taxes.

Your Details

First Name:

Last Name:

Email:

Mobile:

Figure 4.5: Venue booking form

Booking Confirmed!



Computer Lab 2020-03-03 — 2020-03-04

NAME : Piyush Kumar

EMAIL : anon@yopmail.com

MOBILE : 8965452378

Order Reference : 8PGBU4SXFL

DONE

Figure 4.6: Booking confirmation page

Qloapps 1.4.1

IIITM MANIPUR

Quick Access

Explore Qloapps Addons

My shop

Piyush Kumar

Search

Dashboard

Catalog

Orders

Customers

Manage Discounts

Hotel Reservation System

Modules and Services

Localization

Preferences

Advanced Parameters

Administration

Stats

IIITM VENUE MANAGEMENT

Manage Venue

MANAGE VENUE

4

+

↺

↻

⌵

ID

Venue Name

Seats

Price

Q Search

1	Auditorium	250	3500	Edit
2	Computer Lab	100	4000	Edit
3	Guest House	80	4000	Edit
4	Meeting Room	30	2500	Edit

IIITM VENUE MANAGEMENT

Manage Venue

Webkul™ - 0.139s

Contact

Bug Tracker

Forum

Addons

Training

Figure 4.7: Admin panel for venue management

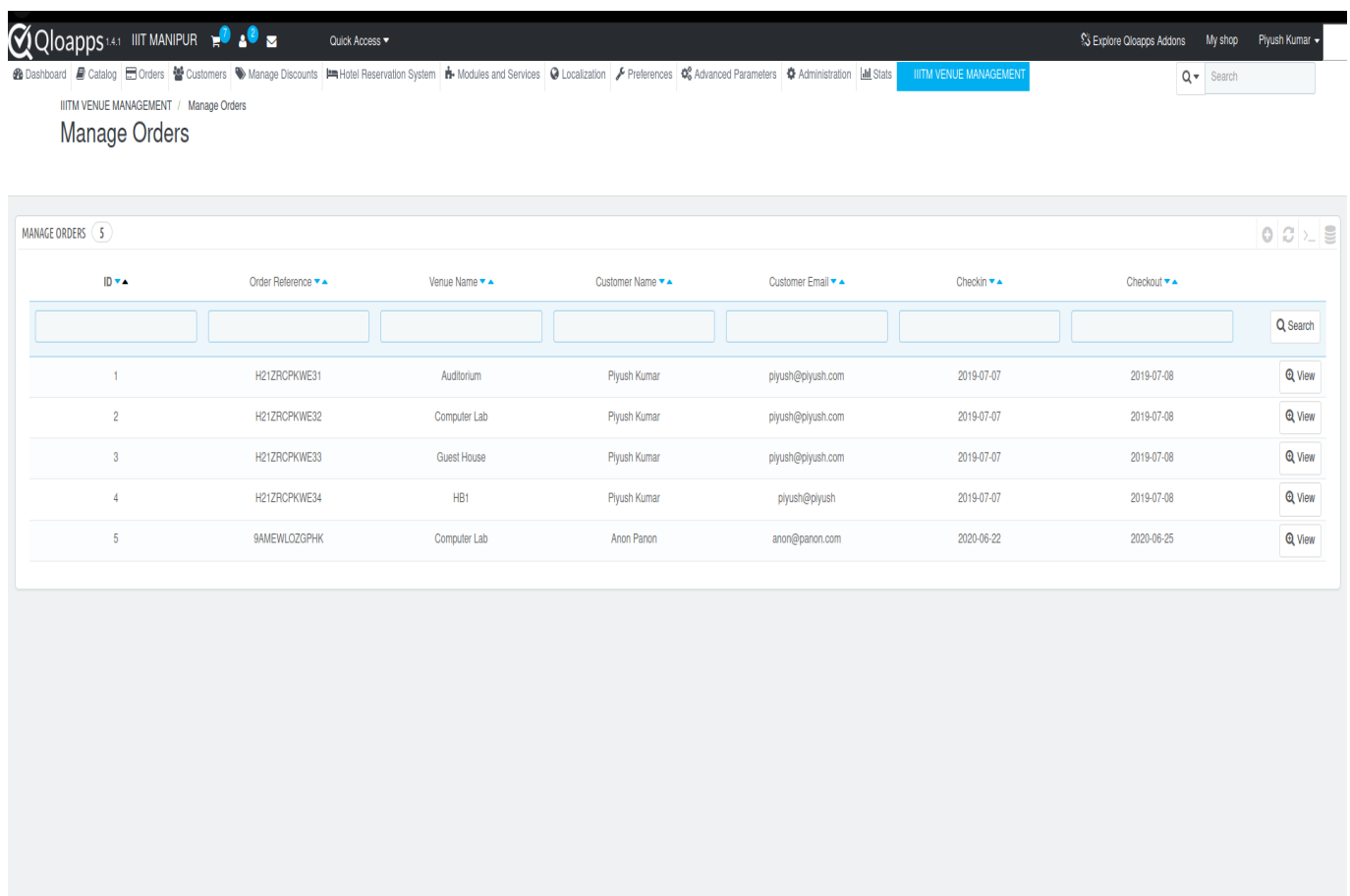


Figure 4.8: Admin Panel for orders page

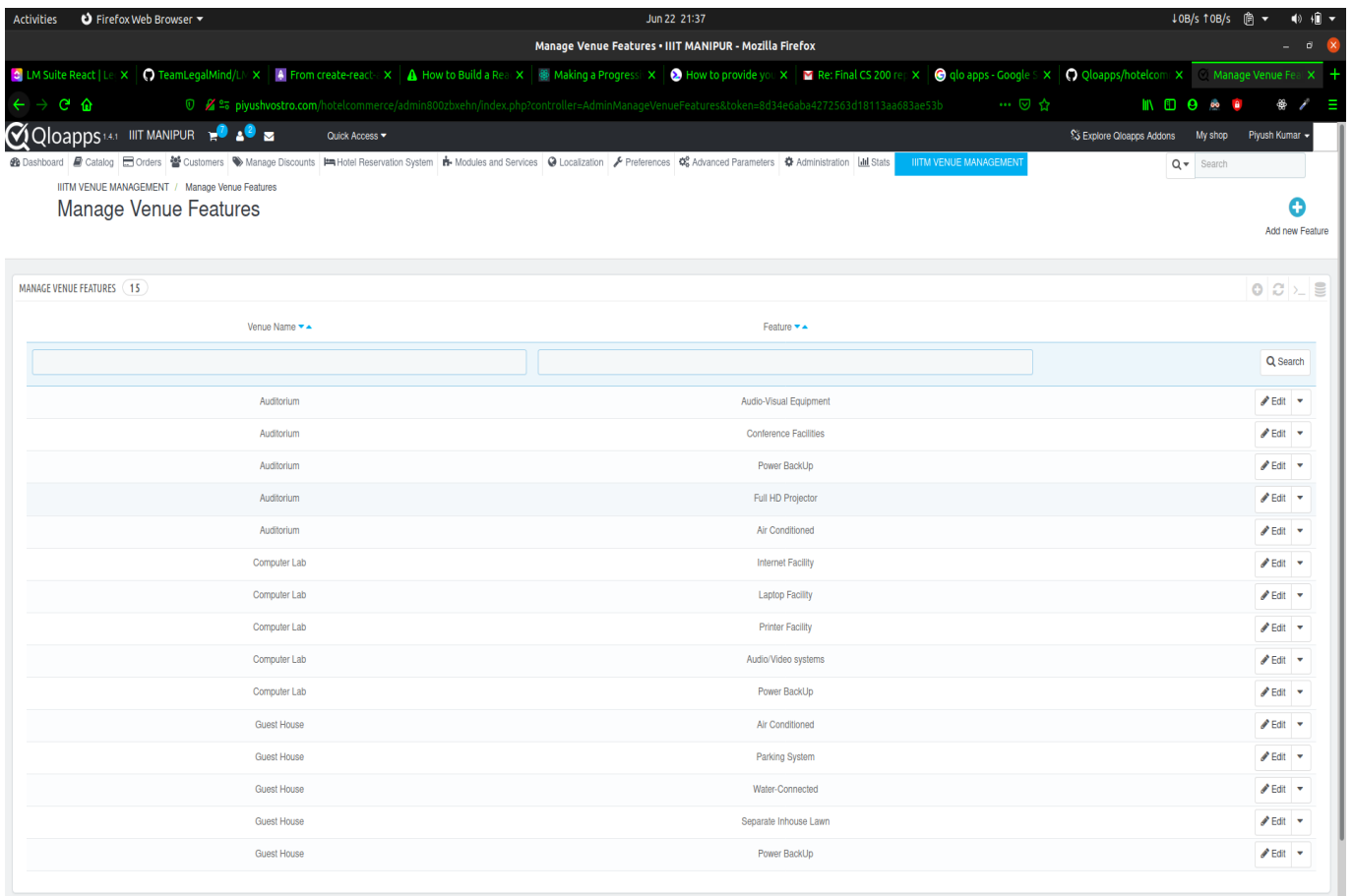


Figure 4.9: Venue features page

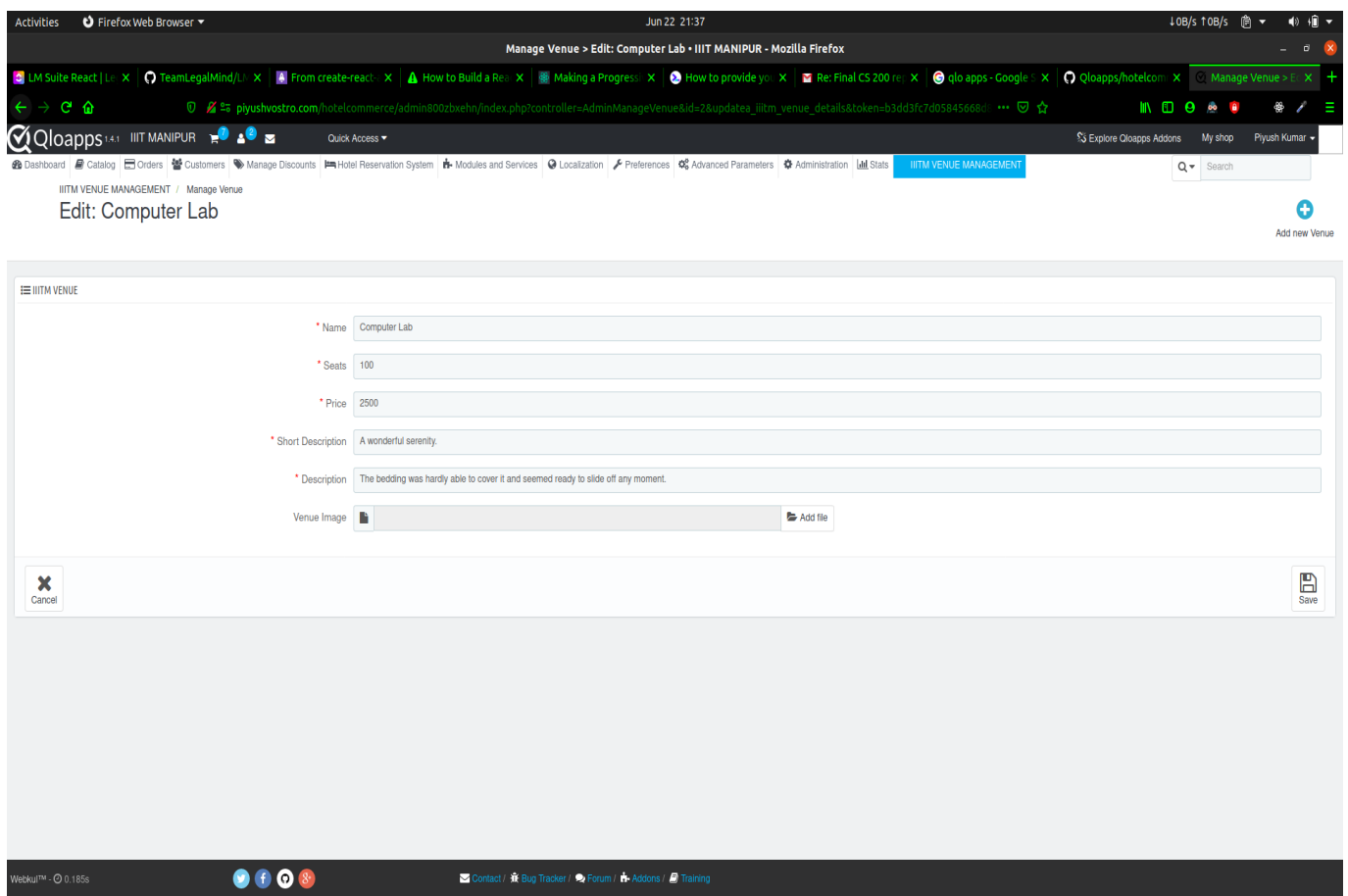


Figure 4.10: Edit existing venue

Chapter 5

Result analysis and Testing

So looking at the final result output images. It looks perfect and feature packed module. Whatever I features or functionalities that I decided to add in my module is added successfully. Every features and functionalities are working upto the mark.

In the homepage, the venue blocks are shown perfectly. Each venue block have a call-to-action button with little description also mentioned in that block. If the call-to-action button is clicked then the user will be redirected to venue details page.

On the venue details page, a image will be shown to identify the venue. All different features of the venue are listed below the venue image. On the right side there are two date pickers which can be used to select checkin and checkout dates and all those will be validated in real time. The price of the venue will also change or udpate in the real time.

When the user clicks on add to cart button after selecting valid checkin and checkout dates, they will be redirected to another page where they will be asked some details in the form. After filling the form successfully, they will reiceve the order successfull page with an unique order-reference ID.

In the admin panel, there is a separate tab for this module under which there are 3 subtabs to manage venue, manage venue features, manage venue orders. From those subtabs admin can easily manage everything they need to manage.They can add new venues with images. After adding new venue they can assign or add features to that venue. They can keep track of each orders made for their venue.

For the case of testing I have tested it manually with lots of testcases and I found some issues that I fixed later on. With all those testing and bug fixing now this module is ready to use and ready to ship.

Chapter 6

Conclusion and Future work

So that's it, I have successfully created a custom module for Qlo to facilitate venue management system and reservation. It was quite enjoyable to work on such a learning project. I got to learn a lot of new things in web development that will help me throughout my whole career.

As a point of future work I can work on an AI bot for our web interface that is capable of serving customers with all the different functionalities of our website in just a click.