18CSC303J- Database management System

Hotel Management System

MINORPROJECT REPORT

Submittedby

S Pranav [RA2011030010212]

Y Sree Santhosh [RA2011030010214]

K Abhiram [RA2011030010218]

UndertheGuidanceof

Dr.N.Krishnaraj

AssociateProfessor,DepartmentofNetworkingandCommunications

Inpartialsatisfactionoftherequirementsforthedegreeof

BACHELOROFTECHNOLOGYinCOMPUTERSCIENCEENGINEERING

WithspecializationinCyberSecurity

SCHOOLOFCOMPUTING

COLLEGE OF ENGINEERING AND

TECHNOLOGYSRMINSTITUTEOFSCIENCEANDTECHNOLOGY

KATTANKULATHUR-603203

May2023

BONAFIDECERTIFICATE

Certified that this project report “Hotel Management System” is the bonafide work of -“Sriramula Pranav [RA2011030010212] , Yandra Santhosh [RA2011030010214] , Kolisetty Abhiram [RA2011030010218]” of III Year/VI Sem B.Tech.(CSE) who carried out the mini project under my supervision for the course 18CSC303J – Database Management Systems in SRM Institute of Science and Technology during the academic year 2022-2023 (Even Sem).

SIGNATURE

FacultyName:Dr.N.Krishnaraj

FacultyDesignation: AssociateProfessor

Annapurani.K

DepartmentName:DepartmentofNetworkingandCommunications

Designation: HOD NWC

Abstract

HOTEL MANAGEMENT SYSTEM

The main artefact of this project is to make a web-based application which will make the management system run smoothly with proper database system. The project allows the hotel employee to have interactive GUI and can even manage room's services and booking in a systematic manner. As the hotel employees are a very busy people and does not have time to plan and manage everything on paper. Now he/she has the power to do it easily with flexible time from this project. The project will allow the hotel management to go on smoothly as it will have features such as Check-in, Check-out, restaurant bills and reservations. As all the important part is stored on the application it is easy to make the bill of a guest and check the availability of the rooms beforehand. The data will be safe with only authorized people having access to the data. The data will also be safe and can be recovered if any natural disaster takes place the data can be restored easily. Restaurant services are also provided here which make it easier when the guest checks out of the hotel. This system is useful both forthe customer and the hotel employee which will have a good effect on both side of.

Table of Contents

List of Figures

Abbreviations

- SQL – Sequential Query Language
- DB – Database
- DBMS – Database Management Systems
- ACID – Atomicity, Consistency, Isolation, Durability
- ER - Entity Relationship (Diagram)

Introduction

Almost all people travel around the world and not all have home around the world so they will prefer to live in hotels. The first thing that come upon peoples mind is if they can secure a room when they reach to their destination. This is a huge task for the staffs in hotel as lot of data of the customers need to be stored properly and efficiently. Implementing a system which can handle all the information about the hotel guest along with the needed facilities all under one system would help the staffs of the hotel. The system will be optimized to add modify the details about customer with what facilities they want to add as food laundry and more. All will be handled by the hotel staff and will be managed well. Humans tend to make simple error and small errors like calculating the total bill of a hotel guest can havea huge impact on small hotels so with the help of my project Kathmandu Prince Hotel will not have such problems and the database will be safe as well.

The utilization of data innovation in the hospitality industry has become enormously in the course of recent years. This voyage has not generally been smooth, yet it has become certain that information technology is presently a basic aggressive weapon in the business. On the chief online portals, OpenTable.com, flaunted that in 2016 dining seats filled in cafés using their online reservation framework surpassed one million. This was a 65% expansion from the earlier year.(Montgomery, 2006) The tourism industry business and voyaging is one of the keyparts of economy in India. Every year a huge number of traveller from each side ofthe globe visits to encounter India. In recent years the innovative headway has soared in India. However not many inns utilize a customized computerized framework to help with a portion of the remaining task at hand. Just bunch of Star lodging has database the board framework for recording data pertinent for the time being and for later in regard to organization and consumer loyalty.

List of General Services and Unique Services

- Identify their relevant risks for the project regarding different factors like social, economic, and environmental.

- Develop a database for storing the guest details and property, employee, and other details.

- Develop a friendly web application which can be usedfor retrieving, insertion.

- Modification and deletion of records stored on database.

- The Website should be able to book the rooms, produce reports using various queries from the database software.

- Analyze different applications like the proposed project tocompare the technology and ideas and benefit from them.

Architecture Diagram

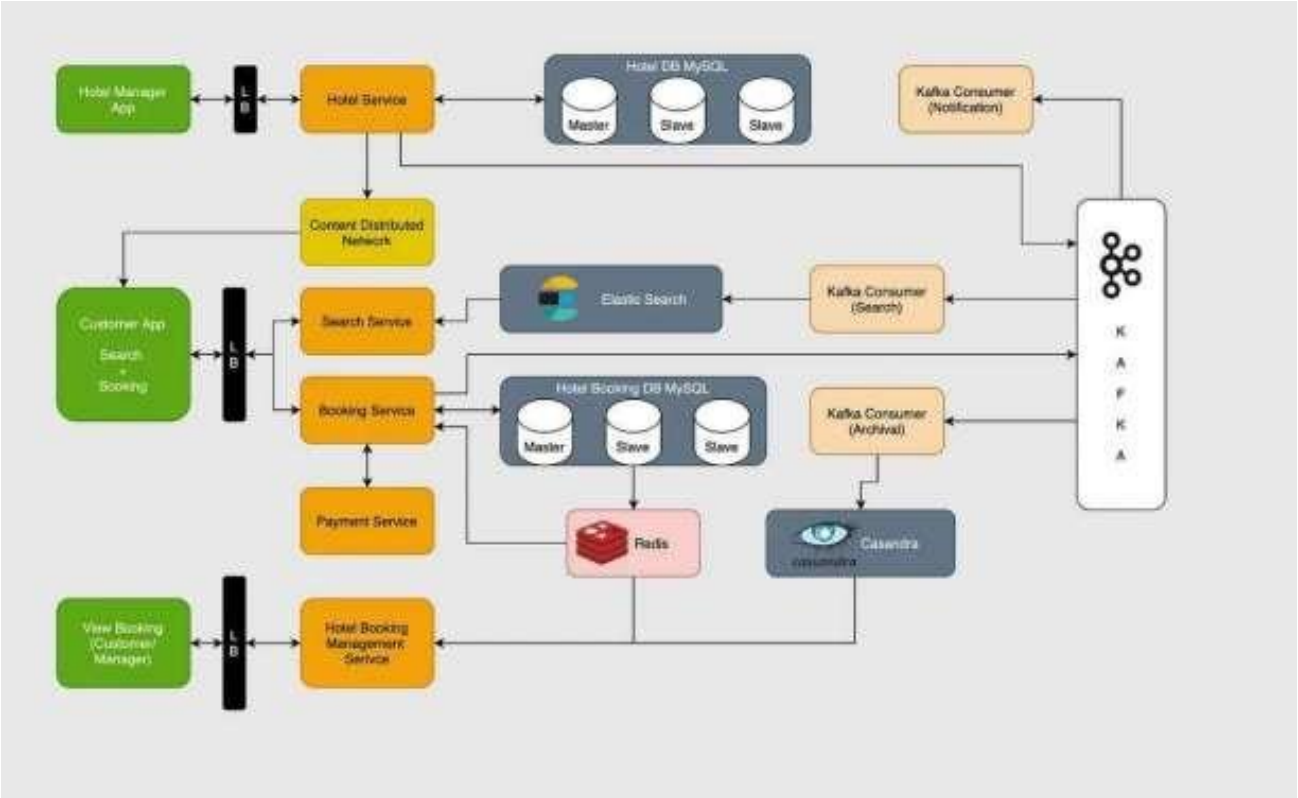

Figure 1: Proposed Architecture Diagram

The Figure 1 shows the high-level architecture of a hotel management system. The system consists of a client application that communicates with a server over HTTP. The server runs a web API that provides access to the system's backend functionality. The backend is responsible for processing requests from the client and interacting with the database.

Use Case Diagram

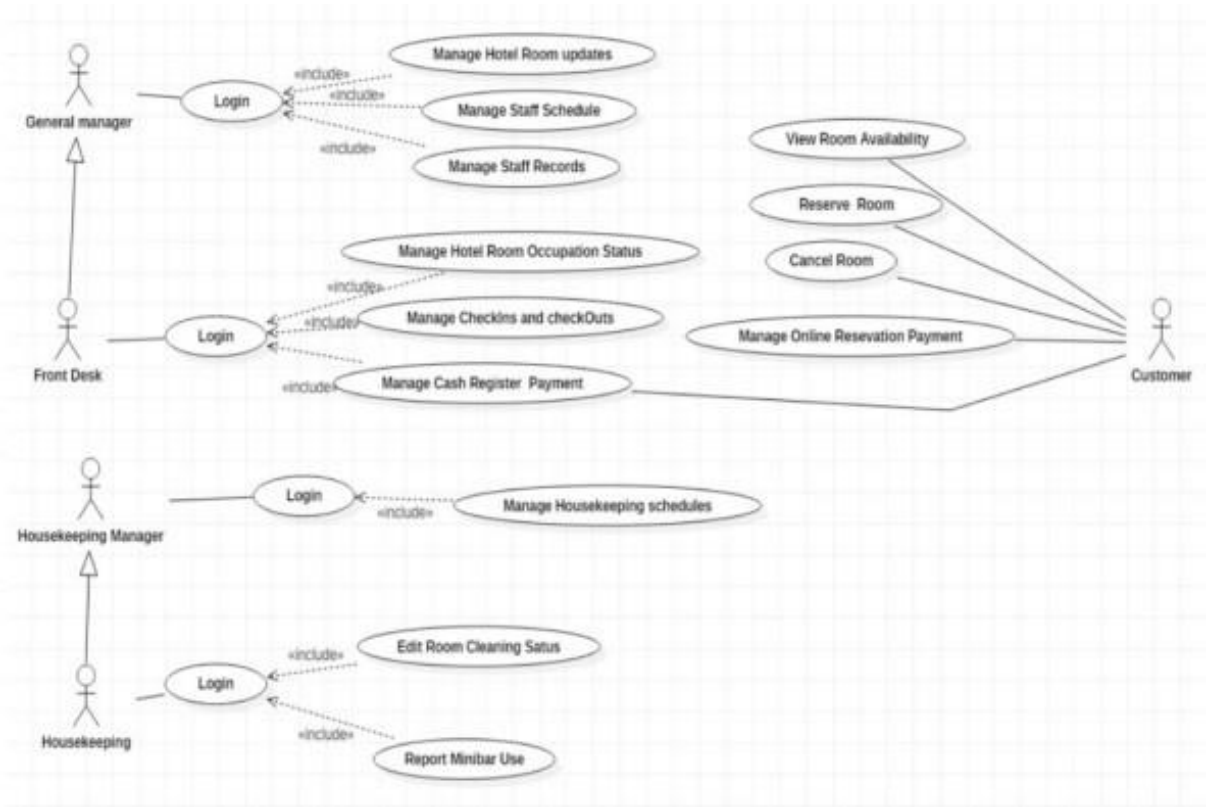

Figure 2: Use Case Diagram

The Figure 2 shows the major use cases of the hotel management system. The diagram alsoshows how these use cases are related to each other through actors and associations.

Entity Relationship Diagram

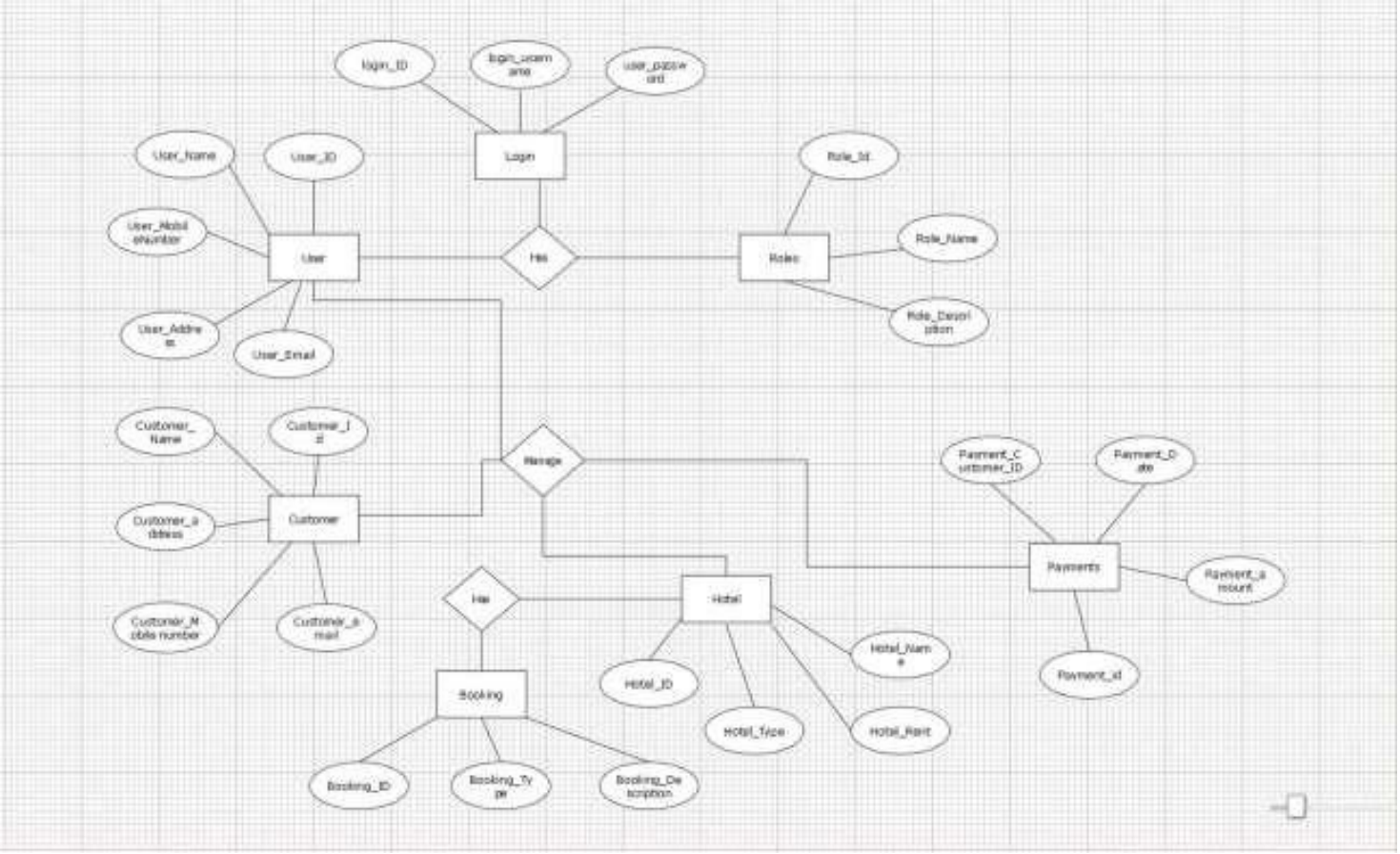

Figure 3: ER Diagram

The Figure 3 shows the entities and relationships involved in a Hotel management system.There are five main entities: User, Customer, Payments, Hotel, Booking.

Frontend (UI) Design and Software used

- HTML:

In essence, HTML is used for creating the primary content of a webpage, giving it structure. You start by writing words, then apply tags or elements to these words. The web browser then reads this and can then understand the heading of a page, any paragraphs, and where the page starts and finishes, thus filling your web page with content. HTML is supported by every single browser and is established on pretty much every webpage in existence. You don't need any licenses, you don't need to pay for it, and it can be pretty easy to learn and code. If we can compare a webpage to the human body, then HTML is the bones of the body.

- CSS:

If HTML is the bones of the body, then CSS is the skin that covers it. It's used for background colour, styling, layout, borders, shadowing – all the essential design bits and bobs that make a webpage look slick and smart. CSS enables you to distinguish between presentation and content by modifying the design and display of HTML elements. Presentation and ease of use are a couple of the main things that CSS has bought to web design by translating the way content looks on a webpage and what else goes on it to complement that content. While frequently used in conjunction with HTML, it is actually independent of it, and can be used with any XML-based markup language.

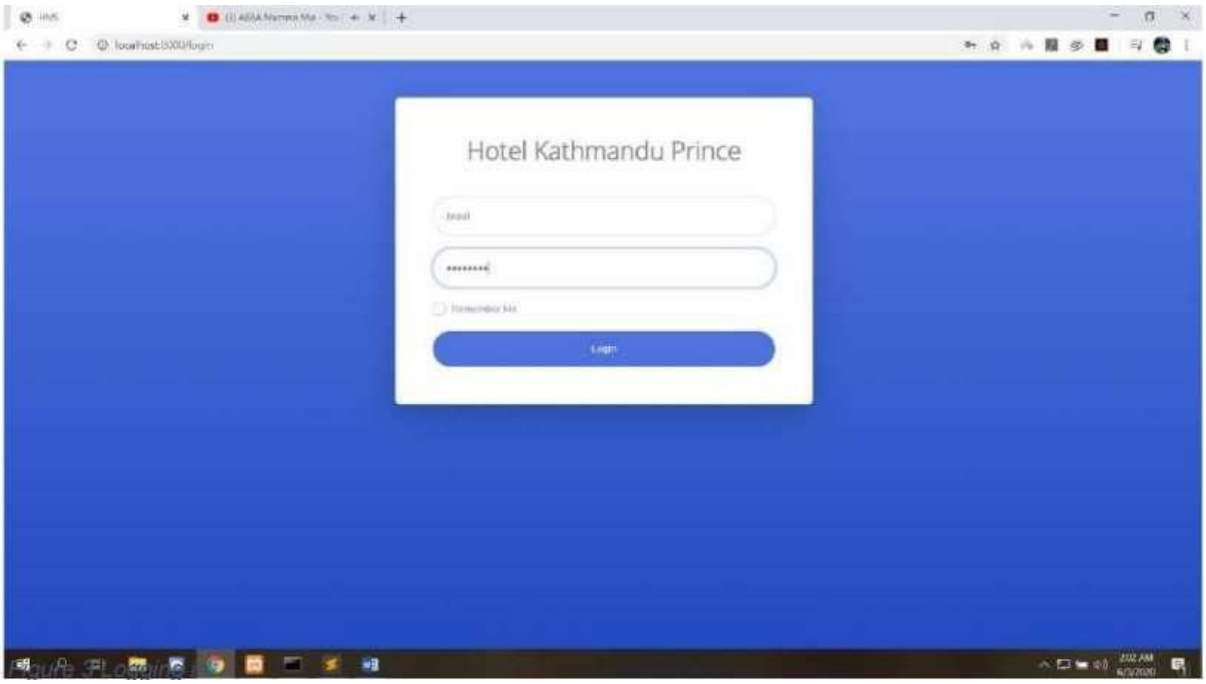

FIGURE 4

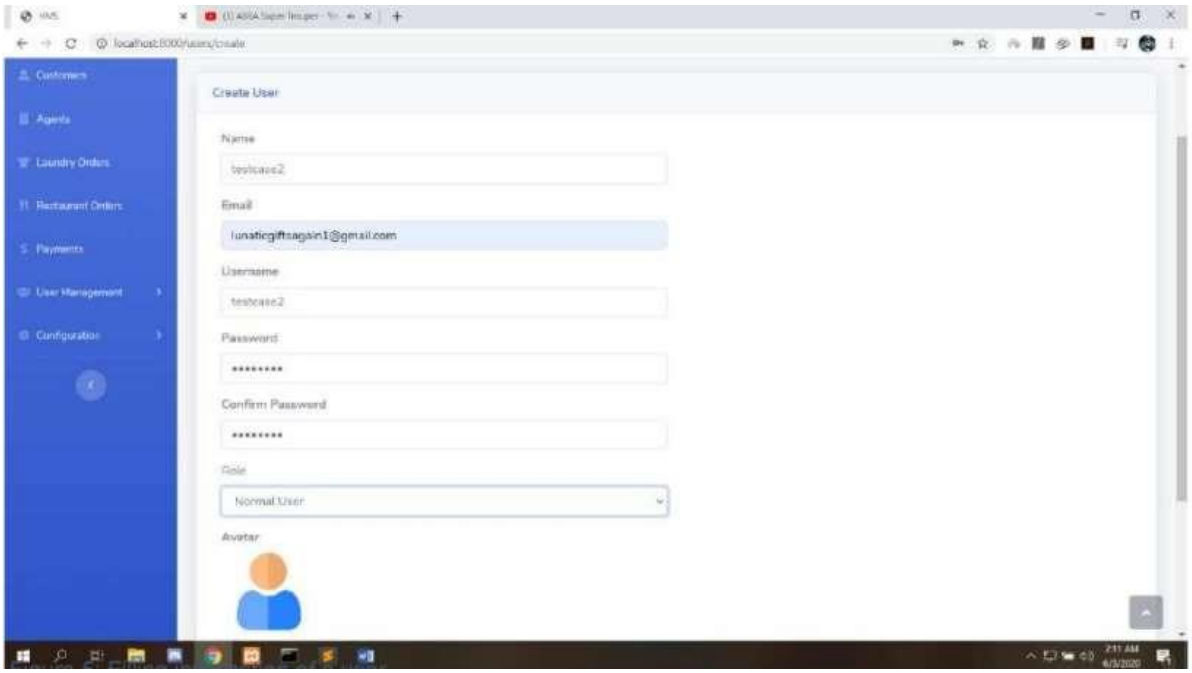

FIGURE 5

Backend (Database) design and Software used

At the point when I chose to utilize PHP as my essential IDE,I considered whereupon database can be utilized with PHP to give it all the more simple and productive touch. From my Research and point by point investigation of the themes I currently realize that Database MYSQL is an imaginative, incredible and instinctive numerous database the executive's apparatus Browse objects, Design tables, Edit columns, Export information and Run questions with a reliable interface. PHP and MYSQL are especially utilized for overseeing database and web applications.

MySQL is described as a free, quick, solid open source social database. It lacks some refinement and offices, however it has a functioning advancement group and, as it goes from discharge to discharge, more abilities are included. At specific occasions there will be an exchange off among speed and capacities, and the MySQL group expect to keep their database motor quick and solid MySQL Advantage is a suite of four open-source programming bundles demonstrated, mainstream, and well-upheld ones–that work together to run intelligent, unique Internet locales. Free-programming ventures that require a full-highlighted database the board framework frequently use MySQL and the arrangement for FYP is to make it as costless as could reasonably be expected. MySQL is additionally utilized in some prominent, huge scale World Wide Web items including, Google and Facebook, Wikipedia its prominence for use with web applications is intently attached to the notoriety of PHP, which is regularly joined with MySQL.

PHP is an open source server-side scripting language that is fundamentally the same as in sentence structure to C language. From this it tends to be inferred three of the essential favourable circumstances of PHP. Right off the bat, it is a cross stage innovation and subsequently PHP applications can be entirely

versatile – depending, obviously, upon any extra segments they are worked to join, for example, seller explicit databases and so on. This conveyability causes an extra advantage by righteousness of the way that most Web facilitating suppliers support PHP, making it genuinely simple to change has if essential. Furthermore, being open source, PHP is always advancing and, all the more significantly, bug fixes are in effect normally actualized deeply libraries, which are uninhibitedly accessible.

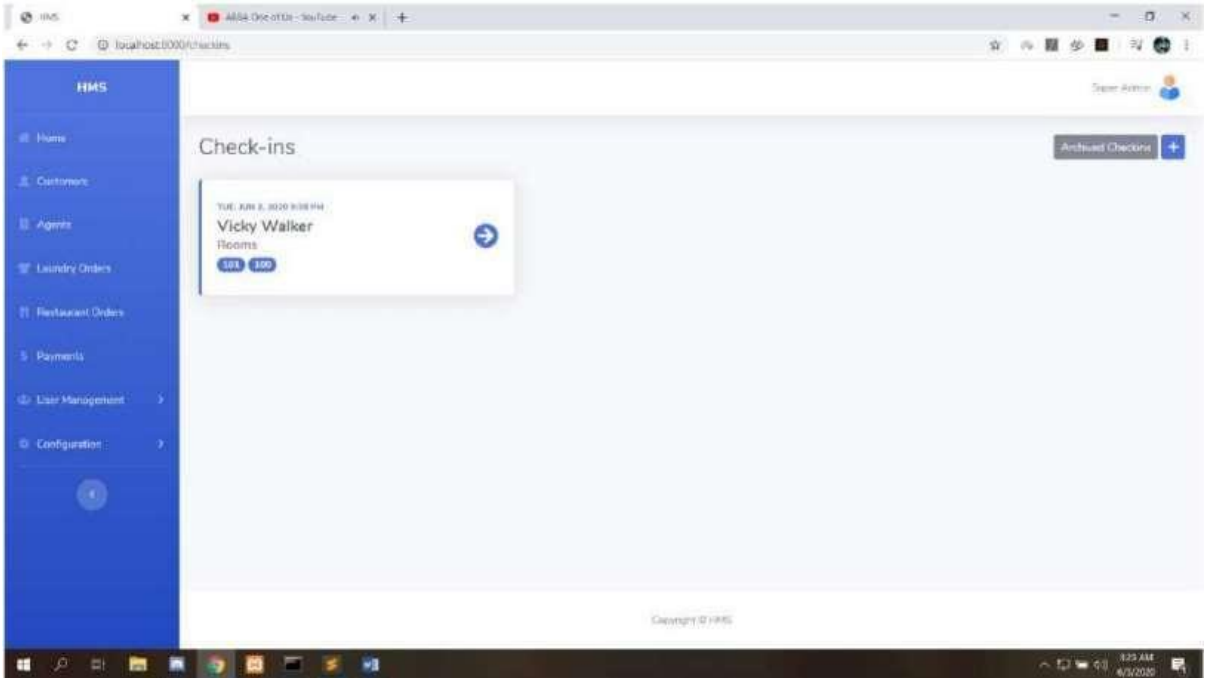

FIGURE 6

Type of connectivity used for DB Access

MySQLi is an API used as a connector function to link the backend of the PHP app to the MySQL database. It works just like the previous version, but it issafer and faster, and provides a better set of functions and extensions. MySQLi was introduced with PHP 5.0.0 and the drivers were installed in 5.3.0. The API

was designed to support MySQL from version 4.1.13 to newer ones. The PHP code consists of a core, with optional extensions to the core functionality. PHP'sMySQL-related extensions, such as the MySQLi extension, and the MySQL extension, are implemented using the PHP extension framework. An extension typically exposes an API to the PHP developer, to allow its facilities to be used programmatically. However, some extensions which use the PHP extension framework do not expose an API to the PHP developer. The PDO MySQL driver extension, for example, does not expose an API to the PHP developer, but provides an interface to the PDO layer above it. MySQLi is an improved version of the older PHP MySQL driver, offering various benefits.

Modules

- Owner Profile: In the Owner profile, he has full access to the system. The owner can view all the details in a graphical way and he has the authority to change the Cost of Room, Room Availability, Service Details, and much more. He can also check the details of the receptionist and staff member currently working at that Hotel. He will get the notification of allthe changes made by the Manager in the system. He will also have special permission to revert those changes if needed. He can also check the transactions made through the day and thought the month and an algorithm will check the progress the Hotel is making.
- Manager Profile: Manager profile can have partial access to the System. The manager can view the availability and can change the cost of the room and other service details. He will also have a special interface where he can manage the staff of the hotel. He can add new people and hecan also remove anyone from the system. And at the same time Owner will receive the notification of these changes.

- Customer Profile: In the Customer profile, people can check the availability of rooms and they can also book a room according to their budget and need. Customer profile contains their name, Contact details, address, and other necessary details, etc. They need to sign-up for booking the hotel, which will make them as well as the manager of the hotel easily interact with each other.
- Staff Profile: In the staff profile, there will be complete details of each staff of the hotel like their personal details and their post at the hotel. They can easily give their attendance to the receptionist and that attendance details will be visible in Realtime to Manager.
- Booking: The customer can easily search their Room from the various option available. This all will be so user- friendly so that Customer will not find any trouble in the booking room. Once the room is searched and the customer finds his choice room then this module helps the customerto book the room by following the rules of the Hotel like Check-in and check-out time of the Hotel, Limit of people per room, etc. If everything is done properly, he will be asked to pay partial or complete money. And then he will be forwarded to Payment Gateway. Where he can easily pay Online.
- Payment Details: After filling in the booking details clients need to pay the money using different options available like cash/net banking/ATM card. The payment portal would use a payment gateway to clear payment. Once payment will clear receipt of payment will be generated automatically and this receipt will be sent to the manager and the Receptionist will get the notification to make the room ready before they arrive.

```
use App\Models\CheckIn;
use
App\Repositories\CheckIn\CheckInRep
ositoryInterface;                    use
App\Repositories\Payment\PaymentRep
ositoryInterface;                    use
App\Services\CheckIn\CheckInServiceI
nterface;
use
App\Services\Payment\PaymentServ
iceInterface; use
App\Models\Payment;
use
Illuminate\Contracts\Foundation\
Application;use
Illuminate\Contracts\View\Facto
ry;
use
Illuminate\Http\Redirect
Response;use
Illuminate\Http\Request;
use Illuminate\Http\Response;
use
Illuminate\Routing\Re
director;use
```

```
Illuminate\View\View;
class PaymentController extends Controller
{
protected
$paymentRepository
;protected
$paymentService;
protected
$checkinRepository;
protected
$checkinService;
/**
* PaymentController constructor.
*
* @param PaymentRepositoryInterface $paymentRepository
* @param PaymentServiceInterface $paymentService
*/
19
4
public function
construct(PaymentRepositoryInterface
$paymentRepository, PaymentServiceInterface
$paymentService, CheckInRepositoryInterface
$checkinRepository, CheckInServiceInterface $checkinService)
{
$this->paymentRepository = $paymentRepository;
```

```
*Display the list of payments.
*
*/
public function index()
{
return view('payments.index', [
'payments' => $this->paymentRepository->all(),
]);
}
/**
* Show the form to add new payment.
*
* @return Application|Factory|View
*/
public function create()
{
19 4
return view('payments.create', [
'checkins' => $this->checkinRepository->query()
->doesntHave('payments')->get(),
]);
}
/**
* Displays the payment addition form with the billing information.
*
* @param CheckIn $checkin
```

```
* @return Application|Factory|View
*/
public function createWithBilling(CheckIn $checkin)
{
return view('payments.create', [
'checkins' => $this-
>checkinRepository->all(),
'selectedCheckin' => $checkin,
'data' => $this->checkinService->getBillAmount($checkin)
]);
}
/**
* Add a new payment.
*
```

```
* @param Request $request
* @param CheckIn $checkin
* @return
Application|RedirectResponse|Red
irector 19 4
*/
public function store(Request $request, CheckIn $checkin)
{
$request->validate([
'discount' =>
'nullable|numeric|min:0|max:100000
0000','payer' =>
'required|in:agent,customer'
]);
return $this->paymentService->create($request)
? redirect(route('payments.index'))->with('success',
'New payment added successfully.')
: back()->with('error', 'New payment could not be added.');
}
public function edit(Request $request, Payment $payment)
{
return view('payments.edit',
[ 'selectedCheckin' => $payment-
>checkin,
'data' => $this->checkinService-
>getBillAmount($payment->checkin),
```

```
'payment' => $payment
]);
}
/**
* Update a specific payment.
*
* @param Request $request
* @param Payment $payment
* @return Payment|RedirectResponse
*/
19
4
public function update(Request $request, Payment $payment)
{
return $this->paymentService->update($request, $payment->id)
? back()->with('success', 'Payment information updated successfully.')
: back()->with('error', 'Failed to update payment information');
}
```

```
/**
* Delete a specific payment.
*
* @param Payment $payment
* @return Application|RedirectResponse|Redirector
*/
public function destroy(Payment $payment)
{
return $this->paymentService->delete($payment->id)
? redirect(route('payments.index'))->with('success',
'Payment deleted successfuly.')
: redirect(route('payments.index'))->with('error', 'Failed to delete payment.');
}
}
```

Restaurant

```
<?php
namespace App\Http\Controllers;
use
App\Http\Requests\Store
Payment;19 4
use
App\Http\Requests\UpdateP
ayment;use
App\Models\CheckIn;
use
App\Repositories\CheckIn\CheckInRep
```

```
ositoryInterface;                use
App\Repositories\Payment\PaymentRep
ositoryInterface;                use
App\Services\CheckIn\CheckInServiceI
nterface;
use
App\Services\Payment\PaymentServ
iceInterface; use
App\Models\Payment;
use
Illuminate\Contracts\Foundation\
Application;use
Illuminate\Contracts\View\Facto
ry;
use
Illuminate\Http\Redirect
Response;use
Illuminate\Http\Request;
use Illuminate\Http\Response;
use
Illuminate\Routing\Re
director;use
Illuminate\View\View;
class PaymentController extends Controller
{
protected
$paymentRepository
```

```
/**
* PaymentController constructor.
*
* @param PaymentRepositoryInterface $paymentRepository
* @param PaymentServiceInterface $paymentService
*/
19
4
public function
construct(PaymentRepositoryInterface
$paymentRepository, PaymentServiceInterface
$paymentService, CheckInRepositoryInterface
$checkinRepository, CheckInServiceInterface $checkinService)
{
$this->paymentRepository = $paymentRepository;
$this->paymentService = $paymentService;
$this->checkinRepository = $checkinRepository;
$this->checkinService = $checkinService;
}
/**
* Display the list of payments.
*
*/
public function index()
{
return view('payments.index', [
```

```
'payments' => $this->paymentRepository->all(),
]);
}
/**
* Show the form to add new payment.
*
* @return Application|Factory|View
*/
public function create()
{
19 4
return view('payments.create', [
'checkins' => $this->checkinRepository->query()
->doesntHave('payments')->get(),
]);
}
```

```
/**
* Displays the payment addition form with the billing information.
*
* @param CheckIn $checkin
* @return Application|Factory|View
*/
public function createWithBilling(CheckIn $checkin)
{
return view('payments.create', [
'checkins' => $this-
>checkinRepository->all(),
'selectedCheckin' => $checkin,
'data' => $this->checkinService->getBillAmount($checkin)
]);
}
/**
* Add a new payment.
*
* @param Request $request
* @param CheckIn $checkin
* @return
Application|RedirectResponse|Red
irector 19 4
*/
public function store(Request $request, CheckIn $checkin)
{
```

```
$request->validate([
'discount' =>
'nullable|numeric|min:0|max:100000
0000','payer' =>
'required|in:agent,customer'
]);
return $this->paymentService->create($request)
? redirect(route('payments.index'))->with('success',
'New payment added successfully.')
: back()->with('error', 'New payment could not be added.');
}
public function edit(Request $request, Payment $payment)
{
return view('payments.edit',
[ 'selectedCheckin' => $payment-
>checkin,
'data' => $this->checkinService->getBillAmount($payment->checkin),
```

```
'payment' => $payment
]);
}
/**
* Update a specific payment.
*
* @param Request $request
* @param Payment $payment
* @return Payment|RedirectResponse
*/
19
4
public function update(Request $request, Payment $payment)
{
return $this->paymentService->update($request, $payment->id)
? back()->with('success', 'Payment information updated successfully.')
: back()->with('error', 'Failed to update payment information');
}
/**
* Delete a specific payment.
*
* @param Payment $payment
* @return Application|RedirectResponse|Redirector
*/
public function destroy(Payment $payment)
{
```

```
return $this->paymentService->delete($payment->id)
? redirect(route('payments.index'))->with('success',
'Payment deleted successfuly.')
: redirect(route('payments.index'))->with('error', 'Failed to delete payment.');
}
}
```

Role

```
<?php
namespace
App\Http\Controllers;
use
App\Forms\Role\Store;
use App\Forms\Role\Update;
use
App\Http\Controllers\Co
ntroller;19 4
use App\Http\Requests\StoreRole;
```

```
use App\Http\Requests\UpdateRole;
use
App\Repositories\Role\RoleReposit
oryInterface; use
App\Services\Role\RoleServiceInter
face;
use App\Models\Role;
use
Illuminate\Auth\Access\Authorizatio
nException;use
Illuminate\Contracts\View\Factory;
use
Illuminate\Http\Redirect
Response;use
Illuminate\Http\Request;
use Illuminate\Http\Response;
use
Illuminate\Routing\Re
director;use
Illuminate\View\View;
use App\Http\Resources\Role as
RoleResource;use
App\Http\Resources\RoleCollecti
on;
class RoleController extends Controller
{
```

```
protected
$roleRepository;
protected
$roleService;
/**
* RoleController constructor.
*
* @param RoleRepositoryInterface $roleRepository
* @param RoleServiceInterface $roleService
*/
public function
construct(RoleRepositoryInterface
$roleRepository, RoleServiceInterface
$roleService)
{
19 4
$this->roleRepository = $roleRepository;
$this->roleService = $roleService;
}
/**
* Display a listing of the role.
*
* @param Request $request
* @return Factory|View
* @throws \Exception
*/
```

```
$search = $request-
>search ?? '';return
view('uac.roles.index'
, [
'roles' => $this->roleRepository->all()
]);
}
/**
* Display the form to create new user.
*
* @return Factory|View
*/
public function create()
{
return
view('components.form.i
ndex',[ 'setup' => (new
Store())->setup()
19 4
]);
}
/**
* Store a newly created role in storage.
*
* @param StoreRole $request
* @return RedirectResponse|Redirector
```

```
*/
public function store(StoreRole $request)
{
return $this->roleService->create($request)
? redirect(route('roles.index'))->with('success', 'New role created successfully.')
: back()->with('error', 'New role could not be created.');
}
/**
* Display the specified role.
*
* @param Role $role
* @return void
*/
public function show(Role $role)
{
abort(404);
}
```

```
/**
19
4
* Display the form to update user.
*
* @param Role $role
* @return Factory|View
*/
public function edit(Role $role)
{
return
view('components.form.i
ndex', [ 'setup' => (new
Update($role))-
>setup()
]);
}
/**
* Update the specified role in storage.
*
* @param UpdateRole $request
* @param Role $role
* @return RedirectResponse|Redirector
*/
public function update(UpdateRole $request, Role $role)
{
```

```
return $this->roleService->update($request, $role->id)
? redirect(route('roles.index'))->with('success', 'Role updated successfully.')
: back()->with('error', 'The role could not be updated.');
}
/**
* Remove the specified role
from storage.19 4
*
* @param Role $role
* @return RedirectResponse|Redirector
*/
public function destroy(Role $role)
{
return $this->roleService->delete($role->id)
? redirect(route('roles.index'))->with('success', 'Role deleted successfully.')
: back()->with('error', 'The role could not be deleted.');
```

```
}
}
Room
<?php
namespace
App\Http\Controllers;
use
App\Forms\Room\Stor
e;
use App\Forms\Room\Update;
use
App\Http\Controllers\Co
ntroller; use
App\Http\Requests\Store
Room; use
App\Http\Requests\Upda
teRoom;
use
App\Repositories\Room\RoomRepos
itoryInterface; use
App\Services\Room\RoomServiceInt
erface;
use App\Models\Room;
use
Illuminate\Auth\Access\Authorizatio
nException;use
```

```
Illuminate\Contracts\View\Factory;
use
Illuminate\Http\Redirect
Response;use
Illuminate\Http\Request;
19 4
use Illuminate\Http\Response;
use
Illuminate\Routing\Re
director;use
Illuminate\View\View;
use App\Http\Resources\Room as
RoomResource;use
App\Http\Resources\RoomCollecti
on;
class RoomController extends Controller
{
protected
$roomRepository;
protected
$roomService;
/**
* RoomController constructor.
*
* @param RoomRepositoryInterface $roomRepository
* @param RoomServiceInterface $roomService
*/
```

```
/**
* Display a listing of the room.
*
* @return Factory|View
* @throws
\Exception19
4
*/
public function index()
{
return
view('components.table.rn',
[ 'records' => $this-
>roomRepository->all(),
'pageTitle' =>
'Configuration',
'tableTitle' =>
'Rooms',
'routePrefix' =>
'rooms','label'
=> 'room'
]);
}
/**
* Display the form to create new user.
*
```

```
* @return Factory|View
*/
public function create()
{
return
view('components.form.i
ndex', [ 'setup' => (new
Store())->setup()
]);
}
/**
* Store a newly created room in storage.
*
* @param
StoreRoom $request
19 4
* @return RedirectResponse|Redirector
*/
public function store(StoreRoom $request)
{
return $this->roomService->create($request)
? redirect(route('rooms.index'))->with('success', 'New room created
```

```
successfully.')
: back()->with('error', 'New room could not be created.');
}
/**
* Display the specified room.
*
* @param Room $room
* @return void
*/
public function show(Room $room)
{
abort(404);
}
/**
* Display the form to update user.
*
* @param Room $room
* @return Factory|View
*/
public function edit(Room $room)
{
19 4
return
view('components.form.i
ndex', [ 'setup' => (new
Update($room))-
```

```
>setup()
]);
}
/**
* Update the specified room in storage.
*
* @param UpdateRoom $request
* @param Room $room
* @return RedirectResponse|Redirector
*/
public function update(UpdateRoom $request, Room $room)
{
return $this->roomService->update($request, $room->id)
? redirect(route('rooms.index'))->with('success', 'Room updated successfully.')
: back()->with('error', 'The room could not be updated.');
```

```
}
/**
* Remove the specified room from storage.
*
* @param Room $room
* @return RedirectResponse|Redirector
*/
public function destroy(Room $room)
{
return $this->roomService-
>delete($room->id) 19 4
? redirect(route('rooms.index'))->with('success', 'Room deleted successfully.')
: back()->with('error', 'The room could not be deleted.');
}
}
```

Room type

```
<?php
namespace
App\Http\Controllers;
use
App\Forms\RoomType\
Store; use
App\Forms\RoomType\
Update; use
App\Http\Controllers\C
ontroller;
```

```
use
App\Http\Requests\StoreRo
omType; use
App\Http\Requests\UpdateR
oomType;
use
App\Repositories\RoomType\RoomTypeRep
ositoryInterface;use
App\Services\RoomType\RoomTypeService
Interface;
use App\Models\RoomType;
use
Illuminate\Auth\Access\Authorizatio
nException;use
Illuminate\Contracts\View\Factory;
use
Illuminate\Http\Redirect
Response;use
Illuminate\Http\Request;
use Illuminate\Http\Response;
use
Illuminate\Routing\Re
director;use
Illuminate\View\View;
use App\Http\Resources\RoomType as
RoomTypeResource;use
App\Http\Resources\
```

```
public function checkout(Request $request, CheckIn $checkin)
{
if($request->has('skipPayment') && $request-
>skipPayment === "true"){ return $this-
>checkinService->checkout($checkin);
}else {
return $this->checkinService->checkoutWithPayment($request, $checkin);
}
}
}
```

19 4

Customer

```
<?php
namespace App\Http\Controllers;
use
App\Http\Requests\StoreCu
stomer; use
App\Http\Requests\Update
Customer;
use
App\Repositories\Customer\CustomerRep
ositoryInterface; use
App\Services\Customer\CustomerService
Interface;
use App\Models\Customer;
use
```

```
Illuminate\Contracts\View
\Factory;                use
Illuminate\Http\RedirectR
esponse;                 use
Illuminate\Http\Request;
use Illuminate\Http\Response;
use
Illuminate\Routing\Redir
ector; use
Illuminate\Support\Faca
des\Auth;use
Illuminate\View\View;
class CustomerController extends Controller
{
protected
$customerRepository
;protected
$customerService;
/**
* CustomerController
constructor. 19 4
*
* @param CustomerRepositoryInterface $customerRepository
* @param CustomerServiceInterface $customerService
*/19 4
/
public function
```

```
{
$this->customerRepository = $customerRepository;
$this->customerService = $customerService;
}
/**
* Display a listing of the customer.
*
* @return Factory|View
*/
public function index()
{
return view('customers.index', [
'customers' => $this->customerRepository->all()
]);
}
/**
* Form to create a new customer.
* Form to create a new customer.
*
* @return Factory|View
*/
public function create()
{
19 4
return view('customers.form');
}
```

```
/**
* Store a newly created customer in storage.
*
* @param StoreCustomer $request
* @return RedirectResponse|Redirector
*/
public function store(StoreCustomer $request)
{
return$this->customerService-
>create($request)19 4 return $this-
>customerService->create($request)
? redirect(route('customers.index'))->with('success',
'New customer created successfully.')
: back()->with('error', 'New customer could not be created.');
```

```
}
/**
* Display the specified customer.
*
* @param Customer $customer
* @return void
*/
public function show(Customer $customer)
{
abort(404);
}
/**
19
4
* Form to update a new customer.
*
* @param Customer $customer
* @return Factory|View
*/
public function edit(Customer $customer)
{
return view('customers.form', compact('customer'));
}
/**
* Update the specified customer in storage.
*
```

```
* @param Request $request
* @param UpdateCustomer $customer
* @return RedirectResponse|Redirector
*/
public function update(UpdateCustomer $request, Customer $customer)
{
return $this->customerService->update($request, $customer->id)
? redirect(route('customers.index'))->with('success',
'Customer updated successfully.')
19 4
: back()->with('error', 'The customer could not be updated.');
}
/**
* Remove the specified customer from storage.
```

```
*
* @param Customer
$customer19 4
* @return RedirectResponse|Redirector
*/
public function destroy(Customer $customer)
{
return $this->customerService->delete($customer->id)
? redirect(route('customers.index'))->with('success',
'Customer deleted successfully.')
: back()->with('error', 'The customer could not be deleted.');
}
}
Food
<?php
namespace
App\Http\Controllers;
use
App\Forms\Food\Store
;
use App\Forms\Food\Update;
use
App\Http\Controllers\C
ontroller;use
App\Http\Requests\Stor
eFood; use
```

```
App\Http\Requests\Upd
ateFood;
use
App\Repositories\Food\FoodReposit
oryInterface; use
App\Services\Food\FoodServiceInte
rface;
use App\Models\Food;
use
Illuminate\Auth\Access\Authorizatio
nException;use
Illuminate\Contracts\View\Factory;
use
Illuminate\Http\Redirect
Response;use
Illuminate\Http\Request;
use Illuminate\Http\Response;
use
Illuminate\Routing\Re
director;19 4
use Illuminate\View\View;
use App\Http\Resources\Food as
FoodResource;use
App\Http\Resources\FoodCollect
ion;
class FoodController extends Controller
{
```

```
/**
* FoodController constructor.
*
* @param FoodRepositoryInterface $foodRepository
* @param FoodServiceInterface $foodService
*/
public function
construct(FoodRepositoryInterface
$foodRepository, FoodServiceInterface
$foodService)
{
$this->foodRepository = $foodRepository;
$this->foodService = $foodService;
}
/**
* Display a listing of the food.
*
* @return Factory|View
* @throws \Exception
*/
public function
index() 19 4
{
return
view('components.table.
ndp', [ 'records' => $this-
```

```
>foodRepository-
>all(), 'pageTitle' =>
'Configuration',
'tableTitle' => 'Menu
Items', 'routePrefix' =>
'foods',
'label' => 'menu item'
]);
}
/**
* Display the form to create new user.
*
* @return Factory|View
*/
public function create()
{
returnview('componentsfor
mindex'19 4return view(
components.form.index ,
[ 'setup' => (new Store())->setup()
```

```
]);
}
/**
* Store a newly created food in storage.
*
* @param StoreFood $request
* @return RedirectResponse|Redirector
*/
19
4
public function store(StoreFood $request)
{
return $this->foodService->create($request)
? redirect(route('foods.index'))->with('success', 'New food created successfully.')
: back()->with('error', 'New food could not be created.');
}
/**
* Display the specified food.
*
* @param Food $food
* @return void
*/
public function show(Food $food)
{
abort(404);
}
```

```
/**
* Display the form to update user.
*
* @param Food $food
* @return Factory|View
*/
public function edit(Food $food)
{
return
view('components.form.i
ndex',
['setup'=>(newUpdate($f
ood))-19 4
[ setup => (new Update($food))
>setup()
]);
}
```

```
/**
* Update the specified food in storage.
*
* @param UpdateFood $request
* @param Food $food
* @return RedirectResponse|Redirector
*/
public function update(UpdateFood $request, Food $food)
{
return $this->foodService->update($request, $food->id)
? redirect(route('foods.index'))->with('success', 'Food updated successfully.')
: back()->with('error', 'The food could not be updated.');
}
/**
* Remove the specified food from storage.
*
* @param Food $food
* @return RedirectResponse|Redirector
*/
public function destroy(Food $food)
{
return $this->foodService->delete($food->id)
? redirect(route('foods.index'))->with('success', 'Food deleted successfully.')
: back()->with('error', 'The food
could not be deleted.'); } }
```

Test Case : Registering user to the system

Test Type Unit Testing Performed Action Filled the information of a new user Expected Outcome . The user profile will be registered and can login

Obtained Outcome the user was able to login Test Result The outcome was a success , Conclusion the action performed is essential to verify if the staff can registered properly.

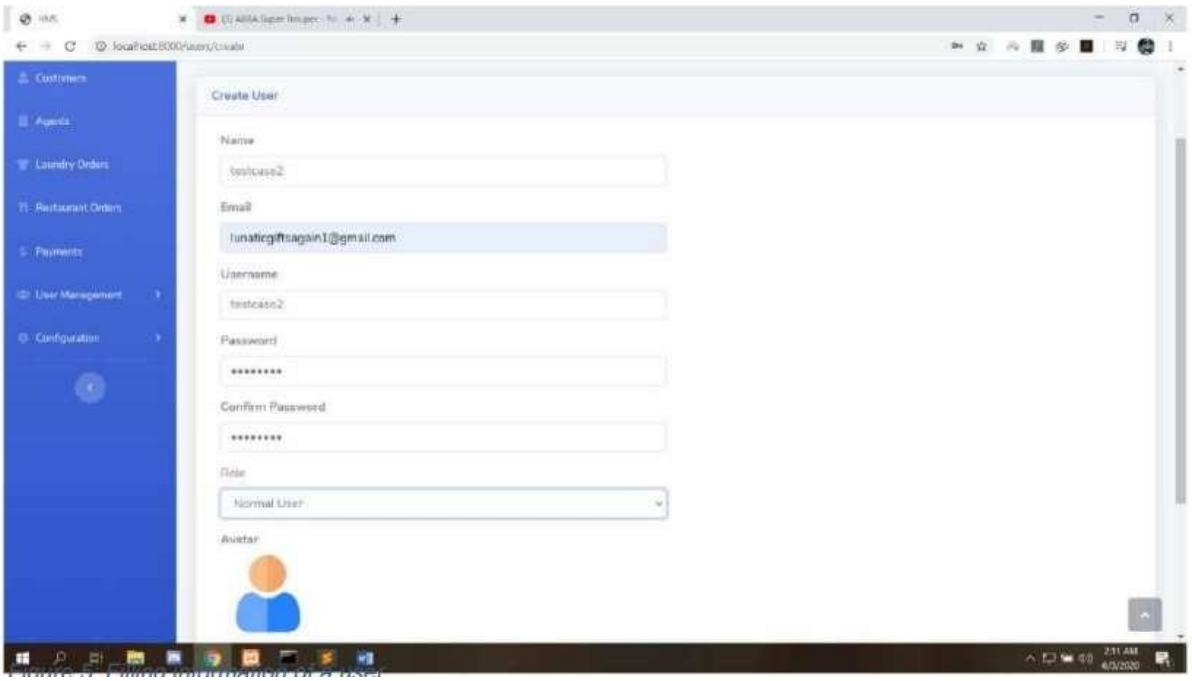

FIGURE 7

Results and Discussions

The system I provided them with has an interface which is easy to use with basic knowledge and can easily use it. There will not be any more calculation errors and the customer's data will be stored safely and can be accessed at the time of need without much time being wasted. Bookings and laundry will be managed without any problem. Humans make errors in their life we are not programmed robots which is always perfect so with the help of the system I have completed its going to be a huge help for the people and my client. Small errors on the bill of a customer during checkout process can make a huge difference. If the customer finds out they have been charged more than they are supposed to pay then they will defame the hotel and many customers will be lost which is a very bad part for the business.

The other major factor if the billing is decreased than it is supposedto then it is going to effect the owner of the hotel as his main source of income will be mismatched and problems might occur and even some misunderstanding can occur between the people which will cause problem on the working environment. Here the system I provided will keep a good track of the money along with a safety with no calculation error and managed system which is the best.

Future updates hold the key for the project to be even more successful with features like advance payments, detailed review reports, user to admin requests, admin to user updates ,high capacity databases, more secured applications, smooth user friendly application etc

Conclusion

A hotel management system is an essential tool for any hotel or hospitality business. It streamlines and automates various processes such as reservations, check-ins, check-outs, billing, inventory management, and more. With a hotel management system in place, hotel staff can provide better service to guests, optimize occupancy rates, reduce errors and fraud, and improve the overall efficiency of the business.

In conclusion, a hotel management system is a crucial investment for any hotel or hospitality business looking to improve its operations and provide better service to its guests. By choosing the right system and implementing it effectively, hotel owners and managers can take their business to new heights and achieve greater success in a highly competitive industry.

References

- Tourism Has generated 20% of total world employment since 2013. Available: Importance of Tourism Industry Grows | .TR (tourism- review.com).

- “Hotel Management and Operations" by Denney G. Rutherford.

- Hotel Management website and the Hotel Management Magazinewebsite (www.hotelmanagement.net)