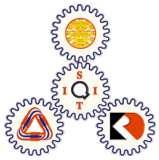


School of Information, Computer and Communication Technology

Sirindhorn International Institute of Technology

ITS322 : Database System

Hotel Management System : CU@MyPlace

By

Kriddanai Roonguthai 5822780334

Prompat Tipphakdee 5822771317

Apiwat Thaiphakdee 5822770467

Wari Maroengsit 5822771333

Semester 1/2017

Abstract

CU@MyPlace is a Hotel Management System that focuses on many general aspects of hotel management such as payment management, staff management, room management and service management. The project will try to use the idea of Relational Database to create the database that could support the system. The project is developed using PHP with mySQL in order to create Web Application that could show General Information about the hotel, Customer System that allows customer to book room and do several functions and Administration System that allows staffs to do management functions. This project simplifies complicating jobs of hotel management into simpler idea that allows us to focus on studying database and system development. Our approach to create this system will involve iterative processes of creating ER Diagrams, creating DataFlow Diagram and creating Context Diagram in order to decide the functions of the database and will be shown more in detail later in this report.

Contents

1 Introduction of The Project **5**
- 1.1 Tourism and Information Technology . . . 5
- 1.2 Hotel Management System . . . 5
- 1.3 What is a good Hotel Management System? . . . 6
- 1.4 Database . . . 7
- 1.5 Database and Hotel Management System . . . 7
- 1.6 Hotel Management System : CU@MyPlace . . . 7

2 Project **8**
- 2.1 ER Diagram . . . 8
- 2.2 Data Flow Diagram . . . 9
- 2.3 Context Diagram . . . 10
- 2.4 Functional Design . . . 11
- 2.5 UI and Test datasets . . . 12

3 Implementation on the Database Management Systems **13**
- 3.1 The System . . . 13
- 3.2 Workflow and Methodology . . . 13
- 3.3 The Database . . . 14
- 3.4 The project:The Test Dataset . . . 16
- 3.5 The project : User Interface . . . 17

4 Explanation on Routines : Procedures and Functions **29**
- 4.1 Procedure 1 : assignroomtobooking . . . 29
- 4.2 Procedure 2 : availableroomtype . . . 29
- 4.3 Procedure 3 : createonlinebooking . . . 30
- 4.4 Procedure 4 : createonlinepayment . . . 30
- 4.5 Procedure 5 : newwalkin . . . 31
- 4.6 Procedure 6 : newwalkinbooking . . . 31
- 4.7 Function 1 : calc_coupon . . . 31

5 Conclusion **33**
5.1 Conclusion on Database Implementation 33
5.2 Conclusion on Web Implementation 34

A Appendix **38**
A.1 SQL Script 38
A.2 Lines count 57
A.3 Github URL 60

List of Figures

2.1 ER Diagram . . . 9
2.2 Data Flow Diagram . . . 10
2.3 Context Diagram . . . 11

3.1 System Detail . . . 13
3.2 Structure of booking table . . . 15
3.3 Structure of message table . . . 15
3.4 Structure of payment table . . . 15
3.5 Structure of request table . . . 16
3.6 Structure of room table . . . 16
3.7 Structure of roomtype table . . . 17
3.8 Structure of message table . . . 17
3.9 Structure of specialoffer table . . . 18
3.10 Structure of staff table . . . 18
3.11 Structure of user table . . . 18
3.12 Structure of usergroup table . . . 18
3.13 Structure of usergroup_has_message table . . . 18
3.14 Test dataset of Booking . . . 19
3.15 Test dataset of Message . . . 19
3.16 Test dataset of Payment . . . 19
3.17 Test dataset of Request . . . 20
3.18 Test dataset of Room . . . 20
3.19 Test dataset of Roomtype . . . 20
3.20 Test dataset of Service . . . 21
3.21 Test dataset of SpecialOffer . . . 21
3.22 Test dataset of Staff . . . 22
3.23 Test dataset of User . . . 22
3.24 Test dataset of Usergroup . . . 23
3.25 Test dataset of Usergroup_has_message . . . 23
3.26 User Interface of Staff Editting . . . 24
3.27 User Interface of Homepage1 . . . 24
3.28 User Interface of Homepage2 . . . 25
3.29 User Interface of Homepage3 . . . 25
3.30 User Interface of login . . . 26

3.31 User Interface of myprofile 26
3.32 User Interface of roomfunction 27
3.33 User Interface of staffdashboard 27
3.34 User Interface of transaction 28
3.35 User Interface of userdashboard 28

5.1 Some views we used in the project 33
5.2 Foreign Key constrains 34
5.3 PHPMyAdmin 35
5.4 Running MySQL in Terminal 35
5.5 MySQL Workbench 36
5.6 The Editor we were using 36
5.7 Github : This is the actual repository of the project 37

Chapter 1

Introduction of The Project

1.1 Tourism and Information Technology

One of the most important part of the economic growth of Thailand is " Tourism ". It provides our country much money that it becomes one of the main income of Thailand. Having brilliant natural resources, kind people, beautiful attraction and good culture are main characteristics of Thailand.

During the stay of tourists, one important part of the stay is the accommodation. There are several choices of accommodation which could be hotel, homestay, resort or even a bangalore. Those accommodations need one same thing is management system.

Since we are moving into the World of Information Technology(IT) , developing a management system for hotel(or any relating accommodations) will be needed to oriented in Information and Technology also. Developing the system and using the idea of IT could bring much more efficiency of resources and generate more income to the hotel. That is why we believe that a good Hotel Management System is very important to any accommodation business anywhere, especially Thailand.

1.2 Hotel Management System

Managing a hotel does not include only booking a room for a guest, but it include so many more tasks which are complicating and hideous job to do manually. This is where Management System will handle the complex part and allow mans to work more efficiently.

Hotel Management System is expertise in management , professional managers, technicians, manuals, systems, etc. On the basis of management fees and share of profits as incentive payment , leading to the prosperity and profit for the hotel.

Our hotel management system will include

1. Hotel Information
    (a) Information about hotel/picture
    (b) Facilities
    (c) Rooms
2. Customer System
    (a) register/login system
    (b) view/edit profile information
    (c) booking system
    (d) request to hotel
    (e) payment
    (f) view the message from admin
3. Hotel Administration System
    (a) login system
    (b) staff and customer profile
    (c) view customer booking
    (d) room management
    (e) message to customer
    (f) payment system
    (g) charge money from customers

And will be demonstrated later in this report.

The term of Hotel Management System includes the management of each and every aspect related to the hotel for the attraction.Smoothness of handling things and proper management such as Booking system , Payment system , Registrations system.The better service can make more attraction for tourist and can make more the profit not only to the hotel but also indirectly to the country.

1.3 What is a good Hotel Management System?

A good Hotel Management System is the system that should be comprehensive of necessary functions that can provide a high quality of service for customer. Also it must allows staffs to use the system efficiently. In other words, the system must accomplish both usability goals and user experience goals.

1.4 Database

A database is a collection of information organized to provide efficient retrieval. The collected information could be in any number of formats (electronic, printed, graphic, audio, statistical, combinations). There are physical (paper/print) and electronic databases. A database could be as simple as an alphabetical arrangement of names in an address book or as complex as a database that provides information in a combination of formats.

Database can generally be looked at as being a collection of records, each of which contains one or more fields (i.e., pieces of data) about some entity (i.e., object), such as a person, organization, city, product, work of art, recipe, chemical, or sequence of DNA. For example, the fields for a database that is about people who work for a specific company might include the name, employee identification number, address, telephone number, date employment started, position and salary for each worker.

1.5 Database and Hotel Management System

By taking advantage of database system we could use it to create complex system for hotel management such as manipulating payments from customer or managing room. Thus running a business of hotel will definitely use database system which will be later demonstrated in this project. Hotel Management System will need a database because to store records such as staff or customers profile, facilities information ,payment records, etc.Also these things which has relation together is a good way because we could use the functions of MySQL to manipulate them.

1.6 Hotel Management System : CU@MyPlace

Our project will create the system of Hotel Management that is capable of handling customers actions and staff actions.

Firstly, the project will include a website that has information of a hotel. We developed it using HTML and CSS. Then the website will brings customer to the booking area which allows user to pick a room and date, then pay for the selection and print confirmation for visiting the hotel.

Second, the customers will have their system for viewing or modifying bookings. They also capable of sending request tickets to hotel for some special preparation.

Third, the administration system will allows staffs to do hotel management duties such as seeing booking, managing status of rooms or contacting customers.

Chapter 2

Project

2.1 ER Diagram

The system includes total 12 entities.

1. booking - keep all booking using FK from other entities
2. message - keep every message
3. payment - keep transactions
4. request - keep every request and the status of it
5. room - represent each instance of a room
6. roomtype - used by room to indicate the type of the room
7. service - keep all service in the hotel
8. specialoffer - keep all coupon number as guest could use it for discount
9. staff - keep all information of every staffs
10. user - keep login info for each user
11. usergroup - keep the group of users
12. usergroup_has_message - records which group has which message - created from M:N Relationship

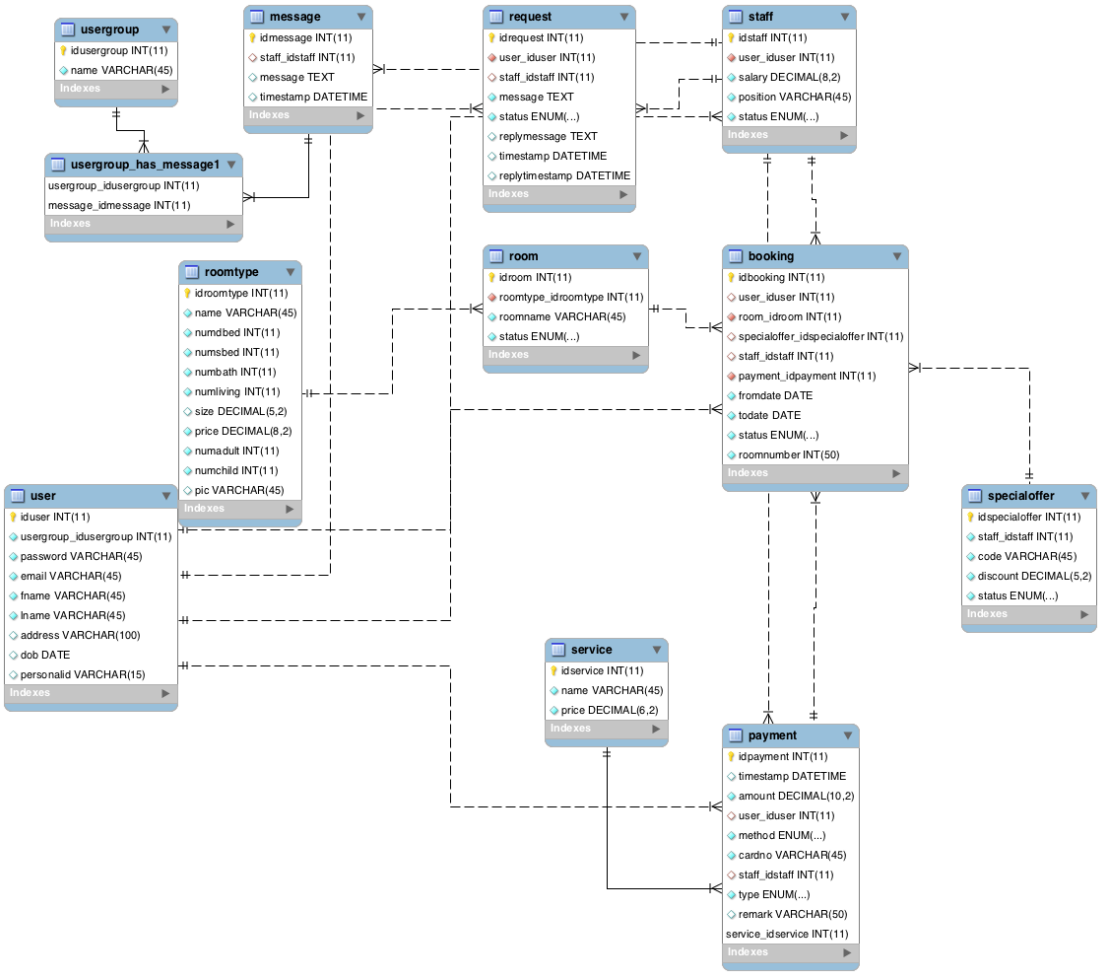


Figure 2.1: ER Diagram

2.2 Data Flow Diagram

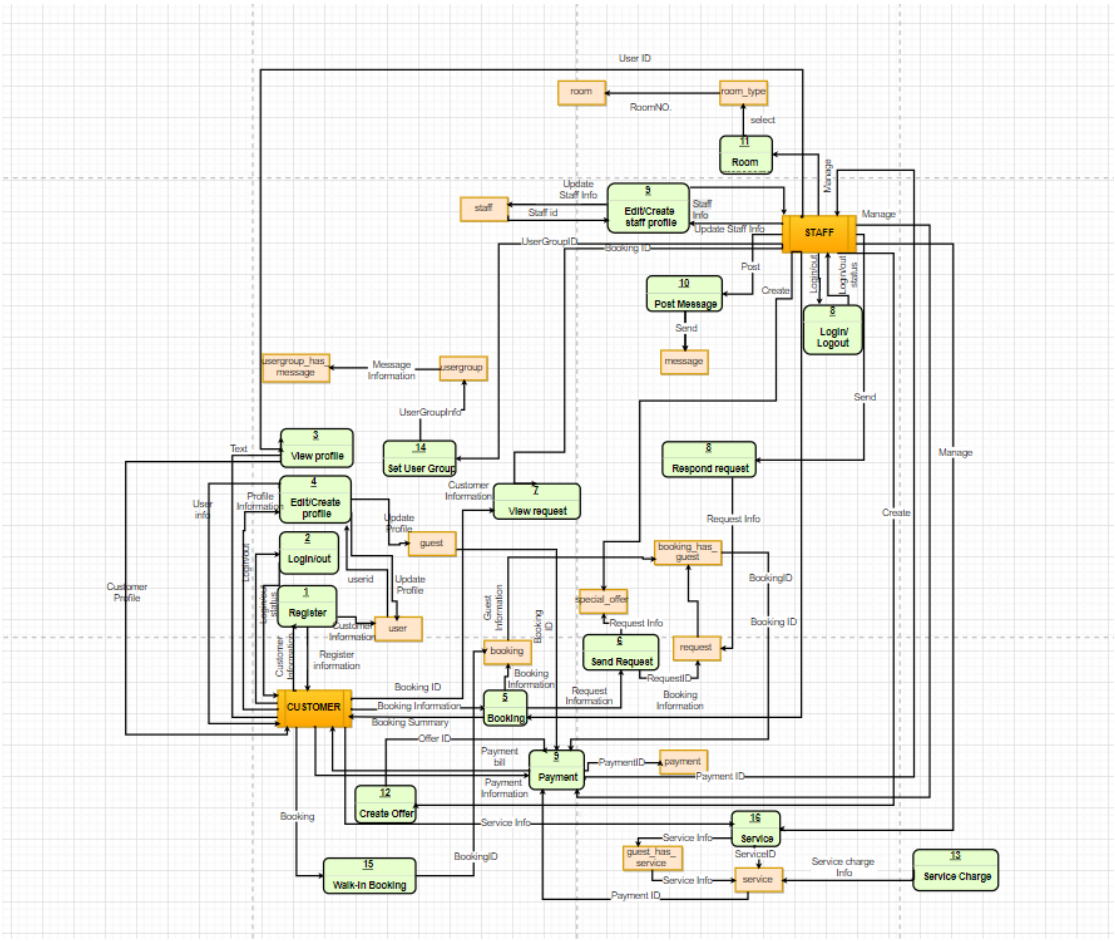


Figure 2.2: Data Flow Diagram

2.3 Context Diagram

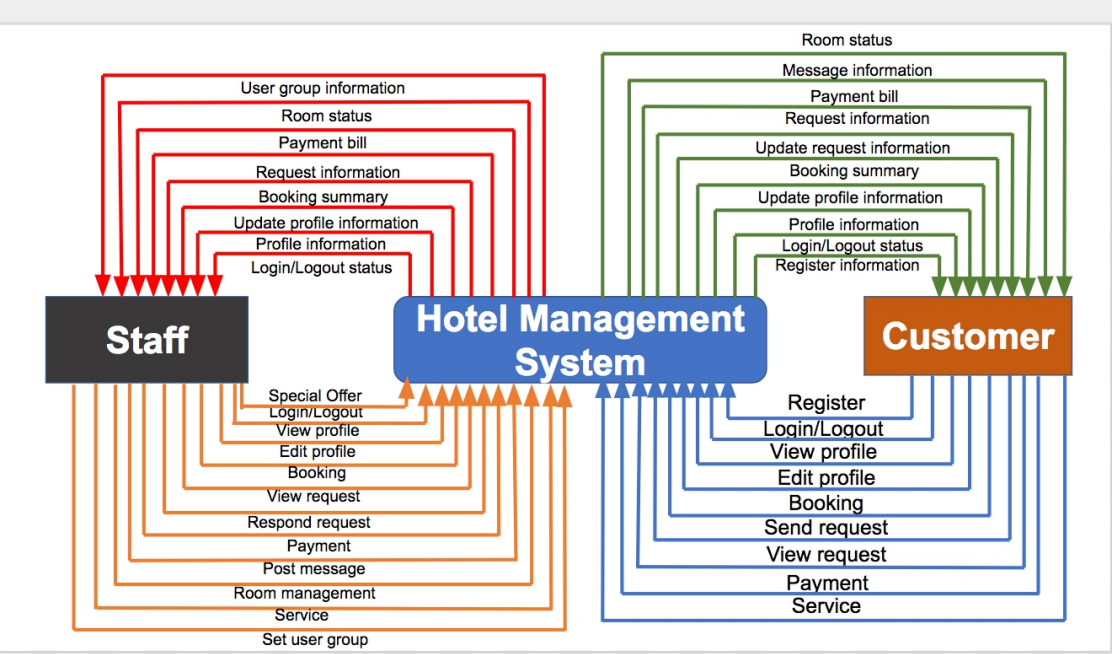


Figure 2.3: Context Diagram

2.4 Functional Design

1. Hotel Information - Main page before login system - this shows pictures of the hotels, general informations of the hotel, booking system interface and portal for login
   (a) Information about hotel/picture
   (b) facilities
   (c) contact us
2. Customer System - Capable of handling customer actions
   (a) register/login system
   (b) view/edit profile information
   (c) booking system
   (d) request to hotel
   (e) payment
   (f) view the message from admin
3. Hotel Administration System - Capable for staff to manage the hotel
   (a) login system

(b) staff and customer profile
(c) view customer booking
(d) room management
(e) message to customer
(f) payment system
(g) charge money from customers

2.5 UI and Test datasets

Images of UI and the test dataset will be discussed in Chapter 4.

Chapter 3

Implementation on the Database Management Systems

3.1 The System

In this project, the Database System we're using is MySQL version 5.6.35 hosted locally on our members' laptops as shown in figure.
The project also implemented purely on PHP7.

3.2 Workflow and Methodology

In order to create continuous flow of the teamworking, team management and planning was very important to us. We used ,somewhat, Agile Philosophy in our development cycles. Our group included development team, which included frontend development and backend development, and testing team.

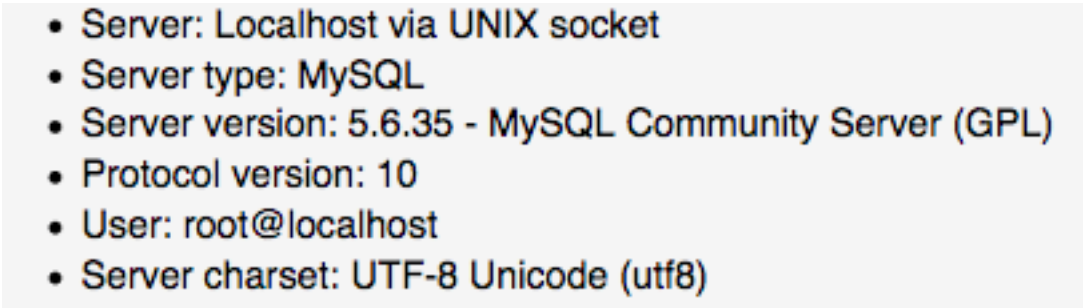


Figure 3.1: System Detail

As we split our implementation time into periods and assign tasks to each team, the flow was very smooth and allowed us to work independently most of the time.

We also used a few tools to help us implement out project more efficiently such as GitHub as Version Control System to keep track of our code and productivity of each team members, We also use GitHub readme as Planning Board for our team to plan future work together as we meet together for daily meeting.

In this project we didnot use any CSS Framework or JS Framework because we believed that learning best should start by learning at the very basic steps. So we implement our own CSS library which isn't Well-Refractored but it is the excellent evidence of our learning.

This reported is written using LaTeX.

3.3 The Database

We planned quite a perfect ER in the very beginning phase of the implementation, but as we continued we need to modify the ER some times because of a few reasons listed below.

1. To support new functional requirements, which are found out while implementing
2. To facilitate physical implementation
3. To do some actions that are constrained by the Database System

The SQL script of our database is exported and included in the appendix. So our final Database includes the following tables.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	idbooking	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	user_iduser	int(11)			Yes	NULL			Change Drop More
3	room_idroom	int(11)			No	None			Change Drop More
4	specialoffer_idspecialoffer	int(11)			Yes	NULL			Change Drop More
5	staff_idstaff	int(11)			Yes	NULL			Change Drop More
6	payment_idpayment	int(11)			No	None			Change Drop More
7	fromdate	date			No	None			Change Drop More
8	todate	date			No	None			Change Drop More
9	status	enum('checked', 'unchecked')	utf8_general_ci		No	None			Change Drop More
10	roomnumber	int(50)			No	None			Change Drop More

Figure 3.2: Structure of booking table

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	idmessage	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique More
2	staff_idstaff	int(11)			Yes	NULL			Change Drop Primary Unique More
3	message	text	utf8_general_ci		Yes	NULL			Change Drop Primary Unique More
4	timestamp	datetime			Yes	NULL			Change Drop Primary Unique More

Figure 3.3: Structure of message table

We also used a lot of routines, namely stored procedure, functions and trigger, and will be discussed later in the next chapter.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	idpayment	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary More
2	timestamp	datetime			Yes	NULL			Change Drop Primary More
3	amount	decimal(10,2)			No	None			Change Drop Primary More
4	user_iduser	int(11)			Yes	NULL			Change Drop Primary More
5	method	enum('cash', 'creditcard')	utf8_general_ci		No	None			Change Drop Primary More
6	cardno	varchar(45)	utf8_general_ci		No	None			Change Drop Primary More
7	staff_idstaff	int(11)			Yes	NULL			Change Drop Primary More
8	type	enum('booking', 'service', '', '')	utf8_general_ci		No	None			Change Drop Primary More
9	remark	varchar(50)	utf8_general_ci		Yes	NULL			Change Drop Primary More

Figure 3.4: Structure of payment table

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**idrequest**	int(11)			No	*None*		AUTO_INCREMENT	Change Drop Primary More
2	**user_iduser**	int(11)			No	*None*			Change Drop Primary More
3	**staff_idstaff**	int(11)			Yes	*NULL*			Change Drop Primary More
4	**message**	text	utf8_general_ci		No	*None*			Change Drop Primary More
5	**status**	enum('open', 'closed')	utf8_general_ci		No	*None*			Change Drop Primary More
6	**replymessage**	text	utf8_general_ci		Yes	*NULL*			Change Drop Primary More
7	**timestamp**	datetime			Yes	*NULL*			Change Drop Primary More
8	**replytimestamp**	datetime			Yes	*NULL*			Change Drop Primary More

Figure 3.5: Structure of request table

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**idroom**	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
2	**roomtype_idroomtype**	int(11)			No	*None*			Change Drop More
3	**roomname**	varchar(45)	utf8_general_ci		No	*None*			Change Drop More
4	**status**	enum('open', 'closed')	utf8_general_ci		No	*None*			Change Drop More

Figure 3.6: Structure of room table

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**idroomtype**	int(11)			No	*None*		AUTO_INCREMENT	Change Drop Primary More
2	**name**	varchar(45)	utf8_general_ci		No	*None*			Change Drop Primary More
3	**numdbed**	int(11)			No	*None*			Change Drop Primary More
4	**numsbed**	int(11)			No	*None*			Change Drop Primary More
5	**numbath**	int(11)			No	*None*			Change Drop Primary More
6	**numliving**	int(11)			No	*None*			Change Drop Primary More
7	**size**	decimal(5,2)			Yes	*NULL*			Change Drop Primary More
8	**price**	decimal(8,2)			No	*None*			Change Drop Primary More
9	**numadult**	int(11)			No	*None*			Change Drop Primary More
10	**numchild**	int(11)			No	*None*			Change Drop Primary More
11	**pic**	varchar(45)	utf8_general_ci		Yes	*NULL*			Change Drop Primary More

Figure 3.7: Structure of roomtype table

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**idservice**	int(11)			No	*None*		AUTO_INCREMENT	Change Drop Primary Unique More
2	**name**	varchar(45)	utf8_general_ci		No	*None*			Change Drop Primary Unique More
3	**price**	decimal(6,2)			No	*None*			Change Drop Primary Unique More

Figure 3.8: Structure of message table

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**idspecialoffer**	int(11)			No	*None*		AUTO_INCREMENT	Change Drop Primary More
2	**staff_idstaff**	int(11)			No	*None*			Change Drop Primary More
3	**code**	varchar(45)	utf8_general_ci		No	*None*			Change Drop Primary More
4	**discount**	decimal(5,2)			No	*None*			Change Drop Primary More
5	**status**	enum('active', 'disabled')	utf8_general_ci		No	*None*			Change Drop Primary More

Figure 3.9: Structure of specialoffer table

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	idstaff	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary More
2	user_iduser	int(11)			No	None			Change Drop Primary More
3	salary	decimal(8,2)			No	None			Change Drop Primary More
4	position	varchar(45)	utf8_general_ci		No	None			Change Drop Primary More
5	status	enum('active', 'nonactive')	utf8_general_ci		No	None			Change Drop Primary More

Figure 3.10: Structure of staff table

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	iduser	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	usergroup_idusergroup	int(11)			No	None			Change Drop More
3	password	varchar(45)	utf8_general_ci		No	None			Change Drop More
4	email	varchar(45)	utf8_general_ci		No	None			Change Drop More
5	fname	varchar(45)	utf8_general_ci		No	None			Change Drop More
6	lname	varchar(45)	utf8_general_ci		No	None			Change Drop More
7	address	varchar(100)	utf8_general_ci		Yes	NULL			Change Drop More
8	dob	date			Yes	NULL			Change Drop More
9	personalid	varchar(15)	utf8_general_ci		Yes	NULL			Change Drop More

Figure 3.11: Structure of user table

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	idusergroup	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary More
2	name	varchar(45)	utf8_general_ci		No	None			Change Drop Primary More

Figure 3.12: Structure of usergroup table

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	usergroup_idusergroup	int(11)			No	None			Change Drop Primary Unique Index More
2	message_idmessage	int(11)			No	None			Change Drop Primary Unique Index More

Figure 3.13: Structure of usergroup_has_message table

idbooking	user_iduser	room_idroom	specialoffer_idspecialoffer	staff_idstaff	payment_idpayment ▲ 1	fromdate	todate	status	roomnumber
1	2	1	*NULL*	*NULL*	1	2017-12-01	2017-12-03	unchecked	1
2	2	4	*NULL*	*NULL*	1	2017-12-01	2017-12-03	unchecked	2
3	2	1	*NULL*	*NULL*	2	2017-12-04	2017-12-06	checked	3
4	2	2	*NULL*	*NULL*	2	2017-12-04	2017-12-06	unchecked	4
5	2	1	*NULL*	*NULL*	3	2017-12-08	2017-12-10	unchecked	5
6	3	5	*NULL*	*NULL*	4	2017-12-01	2017-12-03	unchecked	6

Figure 3.14: Test dataset of Booking

idmessage	staff_idstaff	message	timestamp
1	1	welcome for my hotel	2017-12-13 06:47:13
2	1	welcome for my hotel	2017-12-13 06:47:23
3	1	welcome for my hotel	2017-12-13 06:47:39
4	1	ddd	2017-12-14 03:17:43
5	1		2017-12-14 04:08:06
6	1	Your status is customer	2017-12-14 07:38:41
7	2	Your status is Gold Customer	2017-12-14 07:39:05
8	5	Your status is Platinum Customer	2017-12-14 07:39:40
9	2	Your status is Customer	2017-12-14 07:40:56
10	1	Your status is Gold Customer	2017-12-14 07:41:18

Figure 3.15: Test dataset of Message

3.4 The project:The Test Dataset

the following tables are the test dataset and currently in our database.

idpayment	timestamp	amount	user_iduser	method	cardno	staff_idstaff	type	remark
1	2017-12-13 06:31:50	54000.00	2	creditcard		NULL	booking	NULL
2	2017-12-13 06:36:18	64000.00	2	creditcard		NULL	booking	NULL
3	2017-12-13 06:45:10	40000.00	2	creditcard		NULL	booking	NULL
4	2017-12-14 15:09:55	14000.00	3	creditcard		NULL	booking	NULL
5	2017-12-14 03:19:42	98.00	2	cash		NULL	service	NULL
6	2017-12-14 07:42:20	676.00	2	cash		NULL	service	NULL

Figure 3.16: Test dataset of Payment

idrequest ▲ 1	user_iduser	staff_idstaff	message	status	replymessage	timestamp	replytimestamp
1	2	NULL	Want fruit	closed	Okay	2017-12-13 06:33:23	2017-12-13 06:35:26
2	3	NULL	jfhjf	closed	ooo	2017-12-14 03:11:41	2017-12-14 03:18:38
3	3	NULL	Want more room	closed	Thanks for comment	2017-12-14 07:59:29	2017-12-14 08:00:57
4	2	NULL	Want more facility	open	NULL	2017-12-14 07:59:53	NULL
5	13	NULL	Need more food menu	open	NULL	2017-12-14 08:00:25	NULL

Figure 3.17: Test dataset of Request

idroom	roomtype_idroomtype	roomname	status
1	1	Seaside	open
2	2	Ocean View1	open
3	2	Ocean View2	open
4	3	Supreme1	open
5	3	Supreme2	open
6	3	Supreme3	open
7	2	Golden State	open

Figure 3.18: Test dataset of Room

idroomtype	name	numdbed	numsbed	numbath	numliving	size	price	numadult	numchild	pic
1	Deluxe Pool Access	2	0	2	1	60.00	20000.00	4	2	del1
2	Junior suite	1	1	2	1	45.00	12000.00	3	2	jus1
3	Supreme1D	1	0	1	0	32.00	7000.00	2	1	sup1

Figure 3.19: Test dataset of Roomtype

idservice	name	price
1	American Fried Rice	129.00
2	Pork Steak	199.00
3	Spagetti Cabonara	169.00
4	Beer can	39.00
5	Sandwich	79.00
6	Cheese Burger	99.00
7	Omelette	59.00
8	Tea	25.00
9	Onion Soup	55.00

Figure 3.20: Test dataset of Service

idspecialoffer	staff_idstaff	code	discount	status
1	11	god	25.00	active
2	11	NOCODE	0.00	active

Figure 3.21: Test dataset of SpecialOffer

idstaff	user_iduser	salary	position	status
1	1	98000.00	Managing Director	active
2	4	78000.00	Front Office Manager	active
3	5	74000.00	Assistant Front Office Manager	active
4	6	72000.00	Reservation Manager	active
5	7	71000.00	Chief Telephone operator	active
6	8	68000.00	Receptionist	active
7	9	66000.00	Information Clerk	active
8	10	41000.00	Chief Security	active
9	11	36000.00	Executive Housekeeper	active
10	12	33000.00	Housekeeping Clerk	active

Figure 3.22: Test dataset of Staff

iduser	usergroup_idusergroup	password	email	fname	lname	address	dob	personalid
1	11	staff	staff	staff	staff	123/36	1996-11-01	1151
2	1	user	user	User	User	123/38	1997-02-28	0101
3	2	test	test	test	test	235/423	2017-12-28	0102
4	11	1111	Apiwat@mail.com	Apiwat	Thaiphakdee	123/36	1996-11-01	1152
5	11	1111	Wari@mail.com	Wari	Mareongsit	465/89	1996-10-10	1153
6	11	1111	Kriddanai@hotmail.com	Kriddanai	Roonguthai	169/634	1996-05-22	1154
7	11	1111	Prompat@mail.com	Prompat	Tipphakdee	490/187	1996-12-02	1155
8	11	1111	Jame@mail.com	Jame	Mliner	788/59	1998-02-19	1156
9	11	1111	Joe@mail.com	Joe	Hart	783/121	1994-09-29	1157
10	11	1111	Kevin@mail.com	Kevin	Hoffman	674/27	1990-08-28	1158
11	11	1111	Jane@mail.com	Jane	Molly	599/276	1999-04-02	1159
12	11	1111	Chris@mail.com	Christian	Walker	556/900	1993-07-08	1160
13	1	1234	Benyapa@mail.com	Benyapa	Poonlarp	711/55	1996-12-09	0103

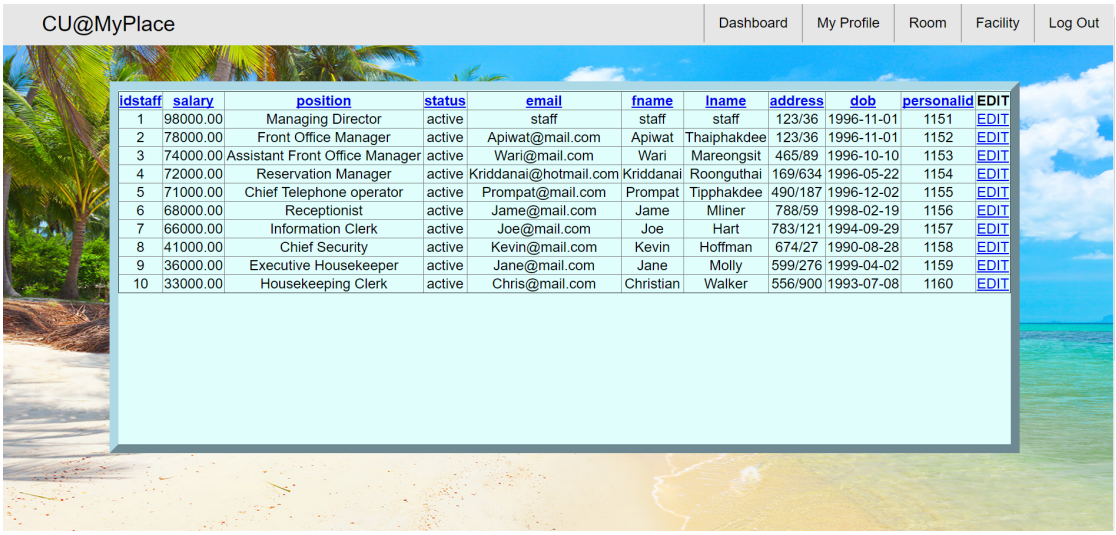
Figure 3.23: Test dataset of User

idusergroup	name
1	customer
2	goldcustomer
3	platinumcustomer
11	staff

Figure 3.24: Test dataset of Usergroup

usergroup_idusergroup	message_idmessage
1	9
2	10
3	8

Figure 3.25: Test dataset of Usergroup_has_message



idstaff	salary	position	status	email	fname	lname	address	dob	personalid	EDIT
1	98000.00	Managing Director	active	staff	staff	staff	123/36	1996-11-01	1151	EDIT
2	78000.00	Front Office Manager	active	Apiwat@mail.com	Apiwat	Thaiphakdee	123/36	1996-11-01	1152	EDIT
3	74000.00	Assistant Front Office Manager	active	Wari@mail.com	Wari	Mareongsit	465/89	1996-10-10	1153	EDIT
4	72000.00	Reservation Manager	active	Kriddanai@hotmail.com	Kriddanai	Roonguthai	169/634	1996-05-22	1154	EDIT
5	71000.00	Chief Telephone operator	active	Prompat@mail.com	Prompat	Tipphakdee	490/187	1996-12-02	1155	EDIT
6	68000.00	Receptionist	active	Jame@mail.com	Jame	Mliner	788/59	1998-02-19	1156	EDIT
7	66000.00	Information Clerk	active	Joe@mail.com	Joe	Hart	783/121	1994-09-29	1157	EDIT
8	41000.00	Chief Security	active	Kevin@mail.com	Kevin	Hoffman	674/27	1990-08-28	1158	EDIT
9	36000.00	Executive Housekeeper	active	Jane@mail.com	Jane	Molly	599/276	1999-04-02	1159	EDIT
10	33000.00	Housekeeping Clerk	active	Chris@mail.com	Christian	Walker	556/900	1993-07-08	1160	EDIT

Figure 3.26: User Interface of Staff Editting

3.5 The project : User Interface

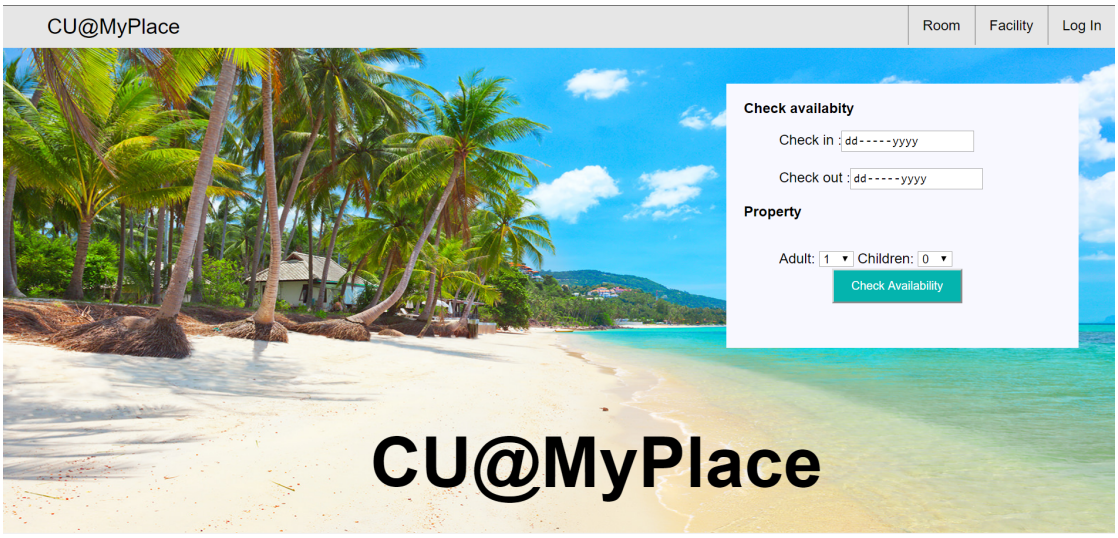


Figure 3.27: User Interface of Homepage1

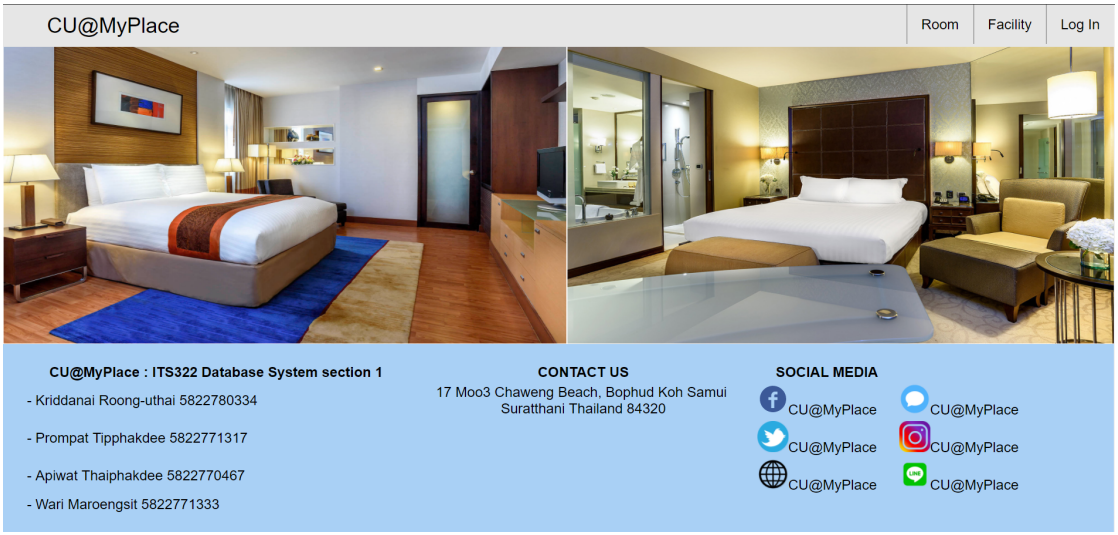


Figure 3.28: User Interface of Homepage2

Figure 3.29: User Interface of Homepage3

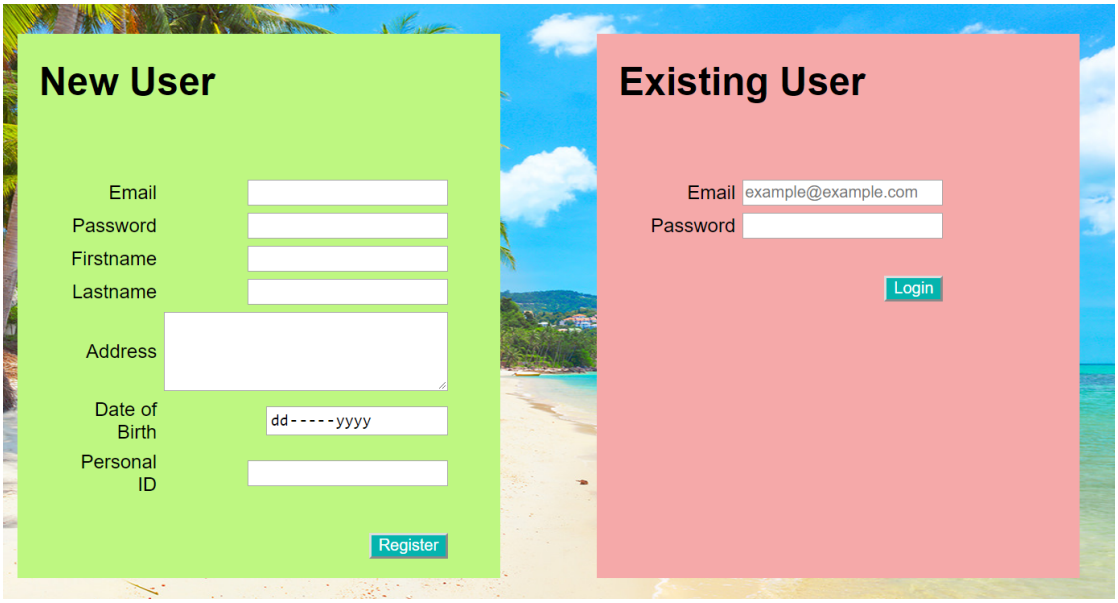


Figure 3.30: User Interface of login

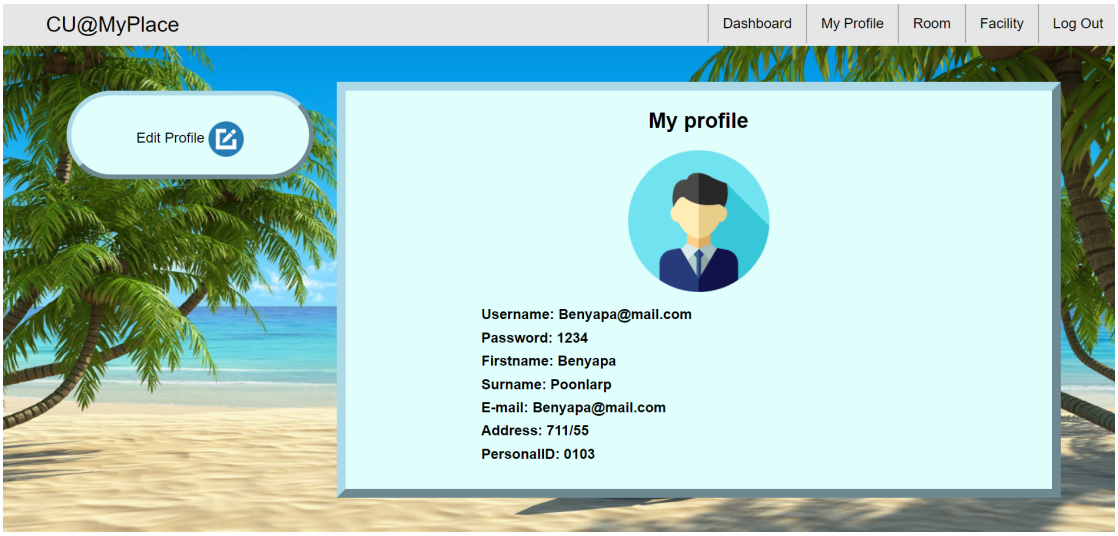


Figure 3.31: User Interface of myprofile

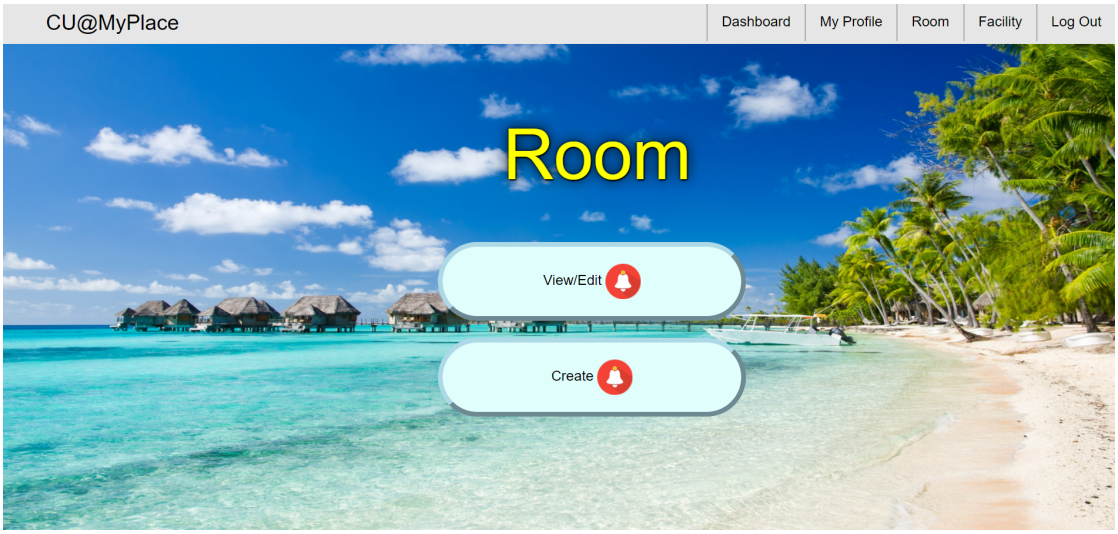


Figure 3.32: User Interface of roomfunction

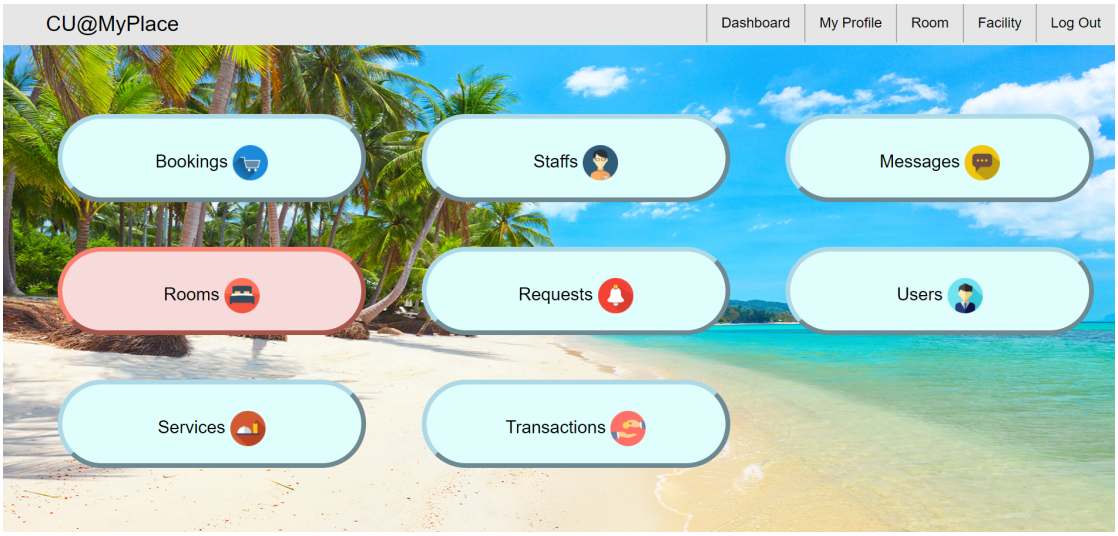
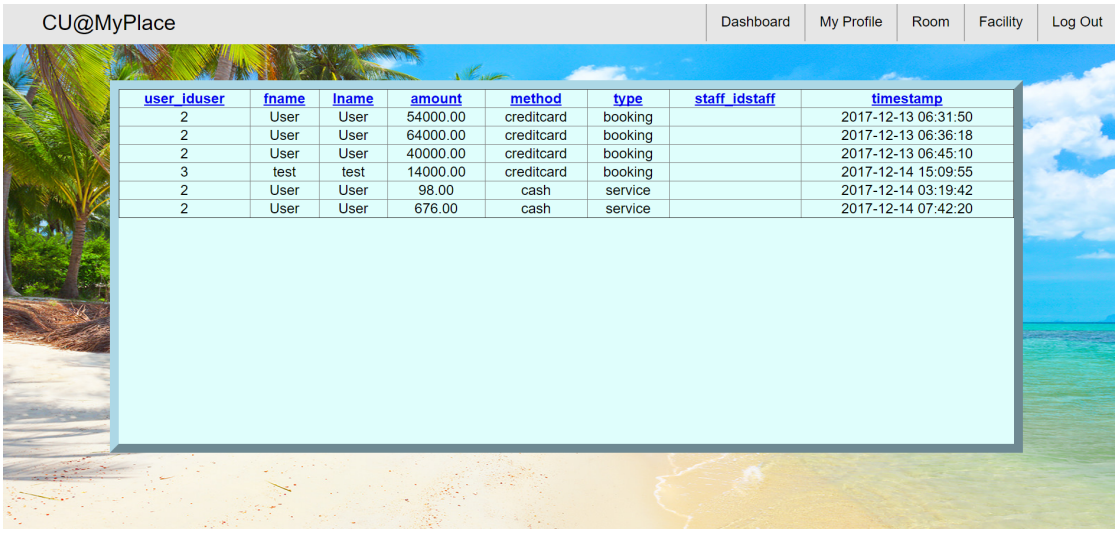


Figure 3.33: User Interface of staffdashboard



user_iduser	fname	lname	amount	method	type	staff_idstaff	timestamp
2	User	User	54000.00	creditcard	booking		2017-12-13 06:31:50
2	User	User	64000.00	creditcard	booking		2017-12-13 06:36:18
2	User	User	40000.00	creditcard	booking		2017-12-13 06:45:10
3	test	test	14000.00	creditcard	booking		2017-12-14 15:09:55
2	User	User	98.00	cash	service		2017-12-14 03:19:42
2	User	User	676.00	cash	service		2017-12-14 07:42:20

Figure 3.34: User Interface of transaction

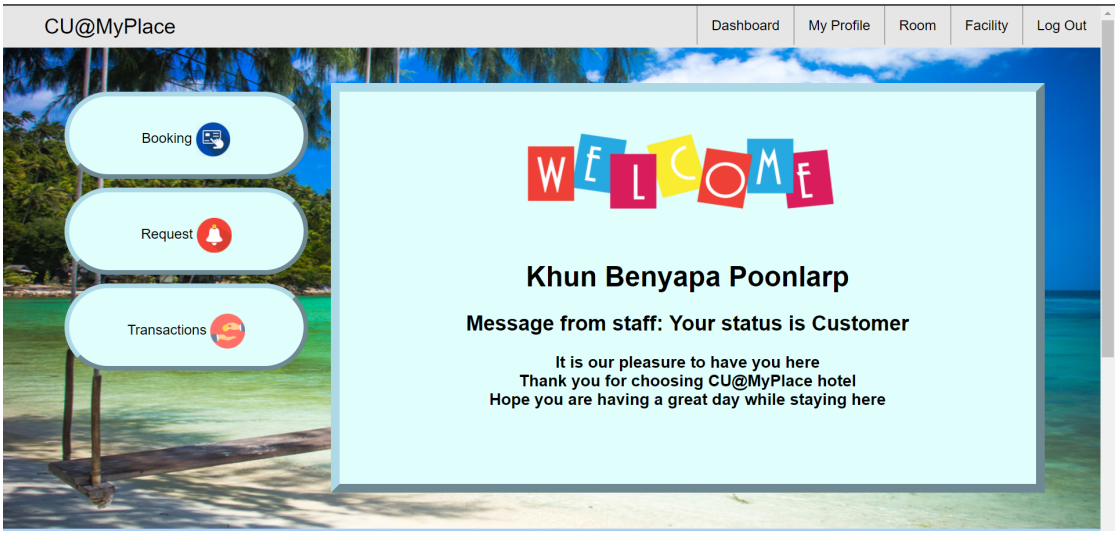


Figure 3.35: User Interface of userdashboard

Chapter 4

Explanation on Routines : Procedures and Functions

4.1 Procedure 1 : assignroomtobooking

```
1
2       DELIMITER $$
3       CREATE DEFINER=`root`@`localhost` PROCEDURE `assignroomtobooking`(
    IN `datein` DATE, IN `dateout` DATE, IN `type` INT, OUT `roomid` INT)
4       BEGIN
5         SELECT room.idroom as roomid FROM room WHERE (room.idroom
6           NOT IN (
7             SELECT booking.room_idroom FROM booking
8             WHERE (booking.fromdate < datein AND booking.todate > datein)
9               OR (booking.fromdate between datein AND dateout - INTERVAL
    1 DAY))
10                AND room.status = 'open' AND room.roomtype_idroomtype =
    type) LIMIT 1;
11      END$$
12      DELIMITER ;
13
```

This procedure is used in booking function where after pressing confirm booking, the procedure will look for an available room and will output the roomid and will be used when insert into the booking table.

4.2 Procedure 2 : availableroomtype

```
1       DELIMITER $$
2         CREATE DEFINER=`root`@`localhost` PROCEDURE `availableroomtype`(
    IN `infromdate` DATE, IN `intodate` DATE)
3         BEGIN
4           SELECT *,COUNT(*) FROM room LEFT JOIN roomtype ON room.
    roomtype_idroomtype = roomtype.idroomtype
5           WHERE (room.idroom NOT IN (
```

```
6           SELECT booking.room_idroom FROM booking WHERE (booking.
    fromdate < infromdate AND booking.todate > infromdate) OR (booking.
    fromdate between infromdate AND intodate - INTERVAL 1 DAY)) AND room.
    status = 'open')
7           GROUP BY roomtype.idroomtype;
8       END$$
9     DELIMITER ;
10
11
```

This procedure return the roomtypes that are available for the customer in the time period which he/she selects.

4.3 Procedure 3 : createonlinebooking

```
1     DELIMITER $$
2     CREATE DEFINER=`root`@`localhost` PROCEDURE `createonlinebooking`
3     (IN `datein` DATE, IN `dateout` DATE, IN `userid` INT, IN `roomid`
    INT, IN `coupon` VARCHAR(50), IN `paymentid` INT, OUT `bookingid` INT)
      BEGIN
          DECLARE couponid INT;
6         SELECT specialoffer.idspecialoffer as couponid FROM
    specialoffer WHERE specialoffer.code = coupon;
7         INSERT INTO `booking` (`idbooking`, `user_iduser`, `
    room_idroom`, `specialoffer_idspecialoffer`, `staff_idstaff`, `
    payment_idpayment`, `fromdate`, `todate`, `status`)
8         VALUES (NULL, userid, roomid, couponid, NULL, paymentid,
    datein, dateout, 'unchecked');
9           SET bookingid = (SELECT LAST_INSERT_ID());
10      END$$
11    DELIMITER ;
12
```

This procedure facilitate storing in the booking table for online booking.

4.4 Procedure 4 : createonlinepayment

```
1     DELIMITER $$
2     CREATE DEFINER=`root`@`localhost` PROCEDURE `createonlinepayment`(
    IN `amount` DECIMAL(10,2), IN `uid` INT, IN `cardno` VARCHAR(50), OUT
    `paymentid` INT)
3     BEGIN
4         INSERT INTO `payment` (`idpayment`, `timestamp`, `amount`, `
    user_iduser`, `method`, `cardno`, `staff_idstaff`,`type`)
5         VALUES (NULL, NOW(), amount, uid, 'creditcard', cardno, NULL,'
    booking');
6         SET paymentid = (SELECT LAST_INSERT_ID());
7       END$$
8     DELIMITER ;
9
```

This procedure store payment table for online booking.

4.5 Procedure 5 : newwalkin

```
1     DELIMITER $$
2     CREATE DEFINER=`root`@`localhost` PROCEDURE `newwalkin`(IN `
    inamount` DECIMAL(10,2), IN `instaffid` INT, IN `incardno` VARCHAR(50)
    , IN `inmethod` VARCHAR(20), OUT `paymentid` INT)
3     BEGIN
4
5         INSERT INTO payment VALUES (NULL, NOW(), inamount, NULL,
    inmethod, incardno, instaffid, 'booking','');
6           SET paymentid = (SELECT LAST_INSERT_ID());
7       END$$
8     DELIMITER ;
9
```

This procedure store payment for walkin booking and is used in the website for creating booking for walkin customers.

4.6 Procedure 6 : newwalkinbooking

```
1     DELIMITER $$
2     CREATE DEFINER=`root`@`localhost` PROCEDURE `newwalkinbooking`(IN `
    infromdate` DATE, IN `intodate` DATE, IN `instaffid` INT, IN `inroomid
    ` INT, IN `offer` VARCHAR(50), IN `paymentid` INT, OUT `bookingid` INT
    )
3     BEGIN
4       DECLARE codeid INT;
5       SELECT specialoffer.idspecialoffer AS codeid FROM specialoffer
    WHERE specialoffer.code = offer;
6       INSERT INTO `booking` (`idbooking`, `user_iduser`, `room_idroom
    `, `specialoffer_idspecialoffer`, `staff_idstaff`, `payment_idpayment
    `, `fromdate`, `todate`, `status`) VALUES (NULL, NULL, inroomid,
    codeid, instaffid, paymentid, infromdate, intodate, 'unchecked');
7       SET bookingid = (SELECT LAST_INSERT_ID());
8     END$$
9     DELIMITER ;
10
```

This procedure store booking for walkin customer and is used in the website for creating booking for walkin customers.

4.7 Function 1 : calc_coupon

```
1     DELIMITER $$
2     CREATE DEFINER=`root`@`localhost` FUNCTION `calc_coupon`(`amount`
    DECIMAL(10,2), `couponcode` VARCHAR(50)) RETURNS decimal(10,2)
    unsigned
```

```
3       BEGIN
4         DECLARE dsc DECIMAL(10,2);
5         SELECT discount into dsc FROM specialoffer WHERE specialoffer.
    code = couponcode;
6         IF (dsc = NULL) THEN
7           return NULL;
8         ELSE
9           SET dsc = amount*(100-dsc)/100;
10            return dsc;
11        END IF;
12      END$$
13      DELIMITER ;
14
```

This function check whether the coupon in valid or not and return discounted amount.

Chapter 5

Conclusion

5.1 Conclusion on Database Implementation

This project makes us realized that ER Database is very useful and very worthful to study because it allows us to design the database using the ER diagram which is very semantic concept. Also learning SQL allows us to send queries to view and edit those tables in the database.

There are a few points, besides query commands, we believed very important to our project.

- Views - Allows us to create a virtual table that could be used for securely views the table, also it facilitate viewing complex join of tables
- Routines - Very important function of SQL because they help us a lot in calling a series of SQL commands without explicitly call all of them
- Foreign Key - Allows data to be verify whether it is correct and relate to the parent table or not.

	Table	Action						Rows	Type	Collation	Size	Overhead
☐	staff_viewbooking	☆	Browse	Structure	Search	Insert	Drop	~0	View	---	-	-
☐	staff_viewmessage	☆	Browse	Structure	Search	Insert	Drop	~0	View	---	-	-
☐	staff_viewrequest	☆	Browse	Structure	Search	Insert	Drop	~0	View	---	-	-
☐	staff_viewroom	☆	Browse	Structure	Search	Insert	Drop	~0	View	---	-	-
☐	staff_viewservice	☆	Browse	Structure	Search	Insert	Drop	~0	View	---	-	-
☐	staff_viewstaff	☆	Browse	Structure	Search	Insert	Drop	~0	View	---	-	-
☐	staff_viewtransaction	☆	Browse	Structure	Search	Insert	Drop	~0	View	---	-	-
☐	staff_viewuser	☆	Browse	Structure	Search	Insert	Drop	~0	View	---	-	-
	8 tables	Sum						~0	InnoDB	utf8_general_ci	0 B	0 B

Figure 5.1: Some views we used in the project

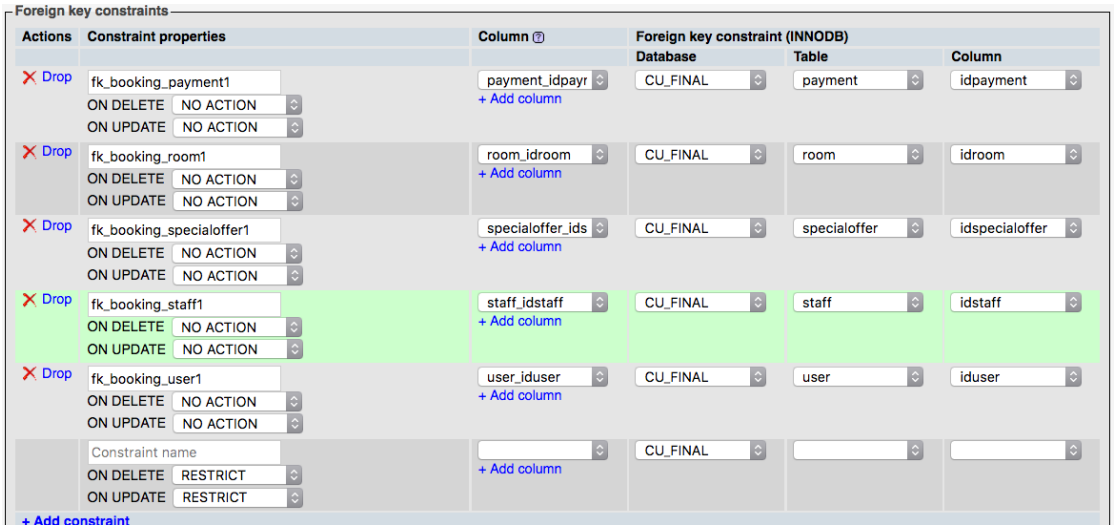


Figure 5.2: Foreign Key constrains

Those points are very important and helped us a lot.

5.2 Conclusion on Web Implementation

We gained a lot of software development experience from working together as it taught us to write thousand lines of code with several weeks working period. We learned to use many tools to develop this project including developing tools such as

- PHPMyAdmin - use for viewing tables in GUI mode.
- MySQL CLI - use for querying and doing works in CLI mode.
- MySQL Workbench - use for designing the DB from ER Diagram.
- Text Editors - Our team use Atom for developing the project
- GitHub - use for version control system

Because of those points mentioned above, We did learned a lot from doing the project. But the project itself still has a lot of room for improvemnt. It still could be developed using CSS Framework for prettier UI or use JavaScript for dynamic webpage for better experience. The system itself still need to be tested in practical use in order to gain more accurate functional requirements and to develop better functions to facilitate hotel's staff and customer jobs which we believed those points could be done if the project is to be used in practice.

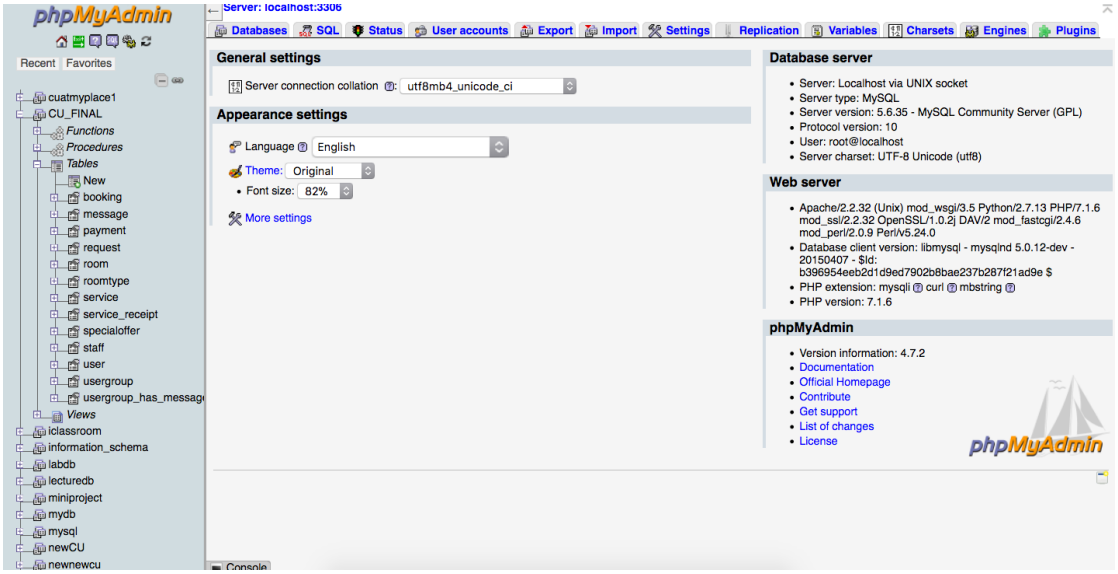


Figure 5.3: PHPMyAdmin

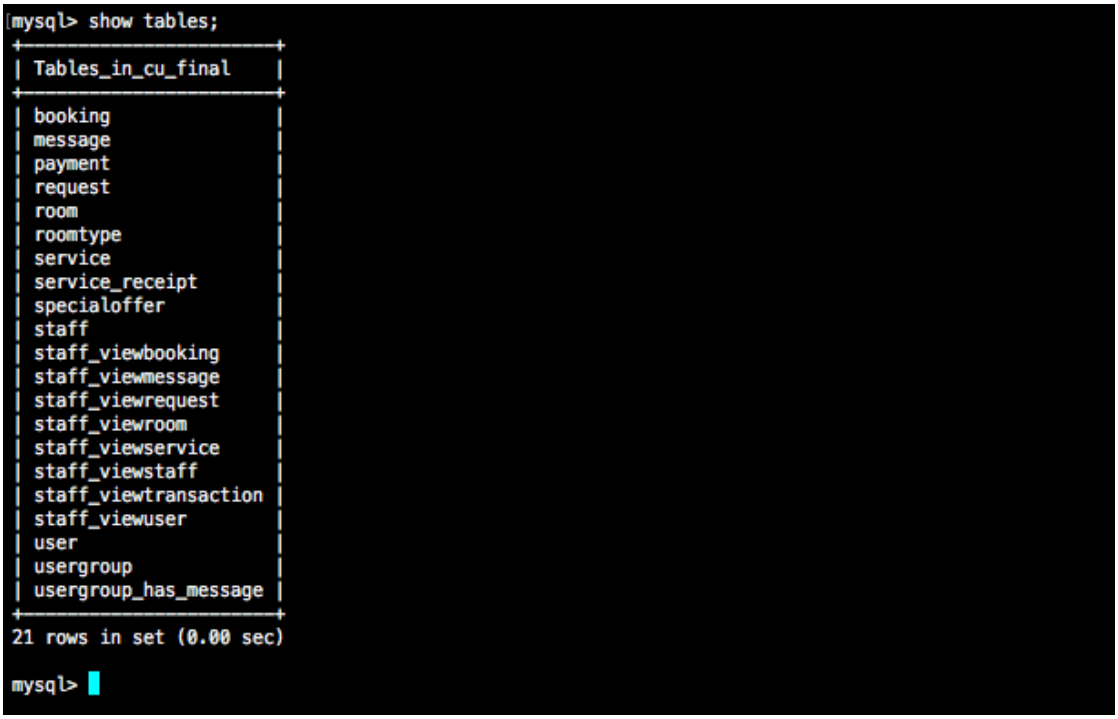


Figure 5.4: Running MySQL in Terminal

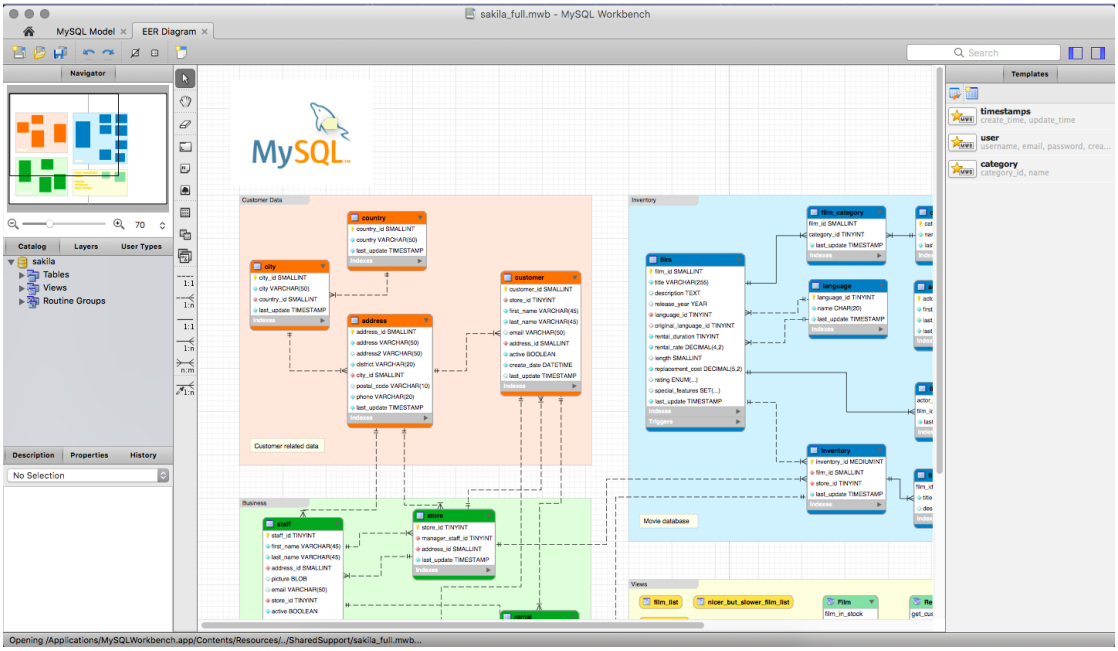


Figure 5.5: MySQL Workbench

Figure 5.6: The Editor we were using

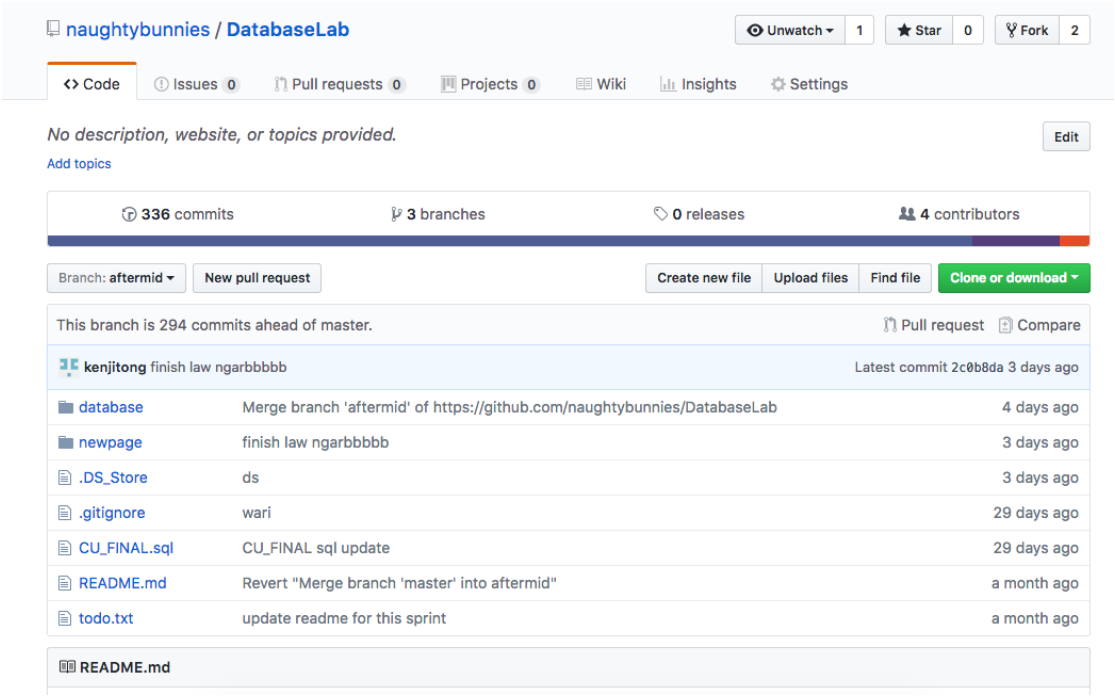


Figure 5.7: Github : This is the actual repository of the project

Appendix A

Appendix

A.1 SQL Script

```
-- phpMyAdmin SQL Dump
-- version 4.7.4
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Generation Time: Dec 14, 2017 at 12:45 PM
-- Server version: 10.1.28-MariaDB
-- PHP Version: 7.1.10

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `cu_final`
--

DELIMITER $$
--
-- Procedures
--
CREATE DEFINER=`root`@`localhost` PROCEDURE `addbooking` (IN `datefrom`
    DATE, IN `dateto` DATE, IN `userid` INT, IN `coupon` INT, IN `roomid`
    INT, OUT `bookingid` INT, OUT `paymentid` INT, IN `inmethod` VARCHAR
    (50), IN `cardno` VARCHAR(50), IN `inamount` DECIMAL, IN `staffid`
    VARCHAR(50))  NO SQL
BEGIN
```

```
31      IF staffid = '' THEN INSERT INTO `payment` (`idpayment`, `timestamp`,
         `amount`, `user_iduser`, `method`, `cardno`, `staff_idstaff`) VALUES
        (NULL, NOW(),inamount, userid, inmethod, cardno, NULL);
32        ELSE INSERT INTO `payment` (`idpayment`, `timestamp`, `amount`, `
        user_iduser`, `method`, `cardno`, `staff_idstaff`) VALUES (NULL, NOW()
        ,inamount, userid, inmethod, cardno, staffid);
33        END IF;
34        SET paymentid = (SELECT LAST_INSERT_ID());
35        IF coupon ='' THEN
36          IF staffid = '' THEN INSERT INTO `booking` (`idbooking`, `
        user_iduser`, `room_idroom`, `specialoffer_idspecialoffer`, `
        staff_idstaff`, `payment_idpayment`, `fromdate`, `todate`, `status`)
        VALUES (NULL, userid, roomid, NULL, NULL, paymentid, datefrom, dateto,
         'unchecked');
37            ELSE INSERT INTO `booking` (`idbooking`, `user_iduser`, `
        room_idroom`, `specialoffer_idspecialoffer`, `staff_idstaff`, `
        payment_idpayment`, `fromdate`, `todate`, `status`) VALUES (NULL,
        userid, roomid, NULL, staffid, paymentid, datefrom, dateto, 'unchecked
        ');
38            END IF;
39         ELSE IF staffid = '' THEN INSERT INTO `booking` (`idbooking`, `
        user_iduser`, `room_idroom`, `specialoffer_idspecialoffer`, `
        staff_idstaff`, `payment_idpayment`, `fromdate`, `todate`, `status`)
        VALUES (NULL, userid, roomid, coupon, NULL, paymentid, datefrom,
        dateto, 'unchecked');
40          ELSE INSERT INTO `booking` (`idbooking`, `user_iduser`, `
        room_idroom`, `specialoffer_idspecialoffer`, `staff_idstaff`, `
        payment_idpayment`, `fromdate`, `todate`, `status`) VALUES (NULL,
        userid, roomid, coupon, staffid, paymentid, datefrom, dateto, '
        unchecked');
41            END IF;
42        END IF;
43        SET bookingid = (SELECT LAST_INSERT_ID());
44    END$$
45
46  CREATE DEFINER=`root`@`localhost` PROCEDURE `assignroomtobooking` (IN `
        datein` DATE, IN `dateout` DATE, IN `type` INT, OUT `roomid` INT)  NO
        SQL
47  BEGIN
48      SELECT room.idroom as roomid FROM room WHERE (room.idroom
49        NOT IN (
50      SELECT booking.room_idroom FROM booking WHERE (booking.fromdate <
        datein AND booking.todate > datein) OR (booking.fromdate between
        datein AND dateout - INTERVAL 1 DAY)) AND room.status = 'open' AND
        room.roomtype_idroomtype = type) LIMIT 1;
51    END$$
52
53  CREATE DEFINER=`root`@`localhost` PROCEDURE `availableroomtype` (IN `
        infromdate` DATE, IN `intodate` DATE)  NO SQL
54  BEGIN
55      SELECT *,COUNT(*) FROM room LEFT JOIN roomtype ON room.
        roomtype_idroomtype = roomtype.idroomtype
56        WHERE (room.idroom
```

```
57        NOT IN (
58      SELECT booking.room_idroom FROM booking WHERE (booking.fromdate <
        infromdate AND booking.todate > infromdate) OR (booking.fromdate
        between infromdate AND intodate - INTERVAL 1 DAY)) AND room.status = '
        open')
59        GROUP BY roomtype.idroomtype;
60    END$$
61
62  CREATE DEFINER=`root`@`localhost` PROCEDURE `createonlinebooking` (IN `
        datein` DATE, IN `dateout` DATE, IN `userid` INT, IN `roomid` INT, IN
        `coupon` VARCHAR(50), IN `paymentid` INT, OUT `bookingid` INT)  NO SQL
63  BEGIN
64      DECLARE couponid INT;
65      SELECT specialoffer.idspecialoffer as couponid FROM specialoffer
        WHERE specialoffer.code = coupon;
66      INSERT INTO `booking` (`idbooking`, `user_iduser`, `room_idroom`, `
        specialoffer_idspecialoffer`, `staff_idstaff`, `payment_idpayment`, `
        fromdate`, `todate`, `status`) VALUES (NULL, userid, roomid, couponid,
         NULL, paymentid, datein, dateout, 'unchecked');
67        SET bookingid = (SELECT LAST_INSERT_ID());
68
69    END$$
70
71  CREATE DEFINER=`root`@`localhost` PROCEDURE `createonlinepayment` (IN `
        amount` DECIMAL(10,2), IN `uid` INT, IN `cardno` VARCHAR(50), OUT `
        paymentid` INT)  NO SQL
72  BEGIN
73      INSERT INTO `payment` (`idpayment`, `timestamp`, `amount`, `
        user_iduser`, `method`, `cardno`, `staff_idstaff`,`type`) VALUES (NULL
        , NOW(), amount, uid, 'creditcard', cardno, NULL,'booking');
74        SET paymentid = (SELECT LAST_INSERT_ID());
75    END$$
76
77  CREATE DEFINER=`root`@`localhost` PROCEDURE `newwalkin` (IN `inamount`
        DECIMAL(10,2), IN `instaffid` INT, IN `incardno` VARCHAR(50), IN `
        inmethod` VARCHAR(20), OUT `paymentid` INT)  NO SQL
78  BEGIN
79
80      INSERT INTO payment VALUES (NULL, NOW(), inamount, NULL, inmethod,
        incardno, instaffid, 'booking','');
81        SET paymentid = (SELECT LAST_INSERT_ID());
82    END$$
83
84  CREATE DEFINER=`root`@`localhost` PROCEDURE `newwalkinbooking` (IN `
        infromdate` DATE, IN `intodate` DATE, IN `instaffid` INT, IN `inroomid
        ` INT, IN `offer` VARCHAR(50), IN `paymentid` INT, OUT `bookingid` INT
        )  NO SQL
85  BEGIN
86      DECLARE codeid INT;
87      SELECT specialoffer.idspecialoffer AS codeid FROM specialoffer WHERE
        specialoffer.code = offer;
88        INSERT INTO `booking` (`idbooking`, `user_iduser`, `room_idroom`, `
        specialoffer_idspecialoffer`, `staff_idstaff`, `payment_idpayment`, `
```

```
    fromdate`, `todate`, `status`) VALUES (NULL, NULL, inroomid, codeid,
    instaffid, paymentid, infromdate, intodate, 'unchecked');
89      SET bookingid = (SELECT LAST_INSERT_ID());
90  END$$
91
92 --
93 -- Functions
94 --
95 CREATE DEFINER=`root`@`localhost` FUNCTION `calc_coupon` (`amount`
    DECIMAL(10,2), `couponcode` VARCHAR(50)) RETURNS DECIMAL(10,2)
    UNSIGNED NO SQL
96 BEGIN
97     DECLARE dsc DECIMAL(10,2);
98     SELECT discount into dsc FROM specialoffer WHERE specialoffer.code =
    couponcode;
99      IF (dsc = NULL) THEN
100       return NULL;
101     ELSE
102       SET dsc = amount*(100-dsc)/100;
103         return dsc;
104     END IF;
105
106  END$$
107
108 DELIMITER ;
109
110 -- --------------------------------------------------------
111
112 --
113 -- Table structure for table `booking`
114 --
115
116 CREATE TABLE `booking` (
117   `idbooking` int(11) NOT NULL,
118   `user_iduser` int(11) DEFAULT NULL,
119   `room_idroom` int(11) NOT NULL,
120   `specialoffer_idspecialoffer` int(11) DEFAULT NULL,
121   `staff_idstaff` int(11) DEFAULT NULL,
122   `payment_idpayment` int(11) NOT NULL,
123   `fromdate` date NOT NULL,
124   `todate` date NOT NULL,
125   `status` enum('checked','unchecked') NOT NULL,
126   `roomnumber` int(50) NOT NULL
127 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
128
129 --
130 -- Dumping data for table `booking`
131 --
132
133 INSERT INTO `booking` (`idbooking`, `user_iduser`, `room_idroom`, `
    specialoffer_idspecialoffer`, `staff_idstaff`, `payment_idpayment`, `
    fromdate`, `todate`, `status`, `roomnumber`) VALUES
134 (1, 2, 1, NULL, NULL, 1, '2017-12-01', '2017-12-03', 'unchecked', 1),
```

```
(2, 2, 4, NULL, NULL, 1, '2017-12-01', '2017-12-03', 'unchecked', 2),
(3, 2, 1, NULL, NULL, 2, '2017-12-04', '2017-12-06', 'checked', 3),
(4, 2, 2, NULL, NULL, 2, '2017-12-04', '2017-12-06', 'unchecked', 4),
(5, 2, 1, NULL, NULL, 3, '2017-12-08', '2017-12-10', 'unchecked', 5),
(6, 3, 5, NULL, NULL, 4, '2017-12-01', '2017-12-03', 'unchecked', 6);

-- --------------------------------------------------------

--
-- Table structure for table `message`
--

CREATE TABLE `message` (
  `idmessage` int(11) NOT NULL,
  `staff_idstaff` int(11) DEFAULT NULL,
  `message` text,
  `timestamp` datetime DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `message`
--

INSERT INTO `message` (`idmessage`, `staff_idstaff`, `message`, `
    timestamp`) VALUES
(1, 1, 'welcome for my hotel', '2017-12-13 06:47:13'),
(2, 1, 'welcome for my hotel', '2017-12-13 06:47:23'),
(3, 1, 'welcome for my hotel', '2017-12-13 06:47:39'),
(4, 1, 'ddd', '2017-12-14 03:17:43'),
(5, 1, '', '2017-12-14 04:08:06');

-- --------------------------------------------------------

--
-- Table structure for table `payment`
--

CREATE TABLE `payment` (
  `idpayment` int(11) NOT NULL,
  `timestamp` datetime DEFAULT NULL,
  `amount` decimal(10,2) NOT NULL,
  `user_iduser` int(11) DEFAULT NULL,
  `method` enum('cash','creditcard') NOT NULL,
  `cardno` varchar(45) NOT NULL,
  `staff_idstaff` int(11) DEFAULT NULL,
  `type` enum('booking','service','','') NOT NULL,
  `remark` varchar(50) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `payment`
--

```

```
INSERT INTO `payment` (`idpayment`, `timestamp`, `amount`, `user_iduser`, `method`, `cardno`, `staff_idstaff`, `type`, `remark`) VALUES
(1, '2017-12-13 06:31:50', '54000.00', 2, 'creditcard', '', NULL, 'booking', NULL),
(2, '2017-12-13 06:36:18', '64000.00', 2, 'creditcard', '', NULL, 'booking', NULL),
(3, '2017-12-13 06:45:10', '40000.00', 2, 'creditcard', '', NULL, 'booking', NULL),
(4, '2017-12-14 15:09:55', '14000.00', 3, 'creditcard', '', NULL, 'booking', NULL),
(5, '2017-12-14 03:19:42', '98.00', 2, 'cash', '', NULL, 'service', NULL);

-- --------------------------------------------------------

--
-- Table structure for table `request`
--

CREATE TABLE `request` (
  `idrequest` int(11) NOT NULL,
  `user_iduser` int(11) NOT NULL,
  `staff_idstaff` int(11) DEFAULT NULL,
  `message` text NOT NULL,
  `status` enum('open','closed') NOT NULL,
  `replymessage` text,
  `timestamp` datetime DEFAULT NULL,
  `replytimestamp` datetime DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `request`
--

INSERT INTO `request` (`idrequest`, `user_iduser`, `staff_idstaff`, `message`, `status`, `replymessage`, `timestamp`, `replytimestamp`) VALUES
(1, 2, NULL, 'Want fruit', 'closed', 'Okay', '2017-12-13 06:33:23', '2017-12-13 06:35:26'),
(2, 3, NULL, 'jfhjf', 'closed', 'ooo', '2017-12-14 03:11:41', '2017-12-14 03:18:38');

-- --------------------------------------------------------

--
-- Table structure for table `room`
--

CREATE TABLE `room` (
  `idroom` int(11) NOT NULL,
  `roomtype_idroomtype` int(11) NOT NULL,
  `roomname` varchar(45) NOT NULL,
  `status` enum('open','closed') NOT NULL
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `room`
--

INSERT INTO `room` (`idroom`, `roomtype_idroomtype`, `roomname`, `status
    `) VALUES
(1, 1, 'Seaside', 'open'),
(2, 2, 'Ocean View1', 'open'),
(3, 2, 'Ocean View2', 'open'),
(4, 3, 'Supreme1', 'open'),
(5, 3, 'Supreme2', 'open'),
(6, 3, 'Supreme3', 'open'),
(7, 2, 'Gold', 'open');

-- --------------------------------------------------------

--
-- Table structure for table `roomtype`
--

CREATE TABLE `roomtype` (
  `idroomtype` int(11) NOT NULL,
  `name` varchar(45) NOT NULL,
  `numdbed` int(11) NOT NULL,
  `numsbed` int(11) NOT NULL,
  `numbath` int(11) NOT NULL,
  `numliving` int(11) NOT NULL,
  `size` decimal(5,2) DEFAULT NULL,
  `price` decimal(8,2) NOT NULL,
  `numadult` int(11) NOT NULL,
  `numchild` int(11) NOT NULL,
  `pic` varchar(45) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `roomtype`
--

INSERT INTO `roomtype` (`idroomtype`, `name`, `numdbed`, `numsbed`, `
    numbath`, `numliving`, `size`, `price`, `numadult`, `numchild`, `pic`)
     VALUES
(1, 'Deluxe Pool Access', 2, 0, 2, 1, '60.00', '20000.00', 4, 2, 'del1'),
(2, 'Junior suite', 1, 1, 2, 1, '45.00', '12000.00', 3, 2, 'jus1'),
(3, 'Supreme1D', 1, 0, 1, 0, '32.00', '7000.00', 2, 1, 'sup1');

-- --------------------------------------------------------

--
-- Table structure for table `service`
--

```

```
CREATE TABLE `service` (
  `idservice` int(11) NOT NULL,
  `name` varchar(45) NOT NULL,
  `price` decimal(6,2) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `service`
--

INSERT INTO `service` (`idservice`, `name`, `price`) VALUES
(1, 'Rice', '49.00');

-- --------------------------------------------------------

--
-- Table structure for table `specialoffer`
--

CREATE TABLE `specialoffer` (
  `idspecialoffer` int(11) NOT NULL,
  `staff_idstaff` int(11) NOT NULL,
  `code` varchar(45) NOT NULL,
  `discount` decimal(5,2) NOT NULL,
  `status` enum('active','disabled') NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `specialoffer`
--

INSERT INTO `specialoffer` (`idspecialoffer`, `staff_idstaff`, `code`, `
    discount`, `status`) VALUES
(1, 11, 'god', '25.00', 'active'),
(2, 11, 'NOCODE', '0.00', 'active');

-- --------------------------------------------------------

--
-- Table structure for table `staff`
--

CREATE TABLE `staff` (
  `idstaff` int(11) NOT NULL,
  `user_iduser` int(11) NOT NULL,
  `salary` decimal(8,2) NOT NULL,
  `position` varchar(45) NOT NULL,
  `status` enum('active','nonactive') NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `staff`
--
```

```
332
333 INSERT INTO `staff` (`idstaff`, `user_iduser`, `salary`, `position`, `
    status`) VALUES
334 (1, 1, '980000.00', 'Head of Hotel', 'active');
335
336 -- --------------------------------------------------------
337
338 --
339 -- Stand-in structure for view `staff_viewbooking`
340 -- (See below for the actual view)
341 --
342 CREATE TABLE `staff_viewbooking` (
343 `idbooking` int(11)
344 ,`user_iduser` int(11)
345 ,`roomnumber` int(50)
346 ,`roomname` varchar(45)
347 ,`discountcode` int(11)
348 ,`staff_idstaff` int(11)
349 ,`payment_idpayment` int(11)
350 ,`fromdate` date
351 ,`todate` date
352 ,`status` enum('checked','unchecked')
353 ,`name` varchar(45)
354 ,`fname` varchar(45)
355 ,`lname` varchar(45)
356 );
357
358 -- --------------------------------------------------------
359
360 --
361 -- Stand-in structure for view `staff_viewmessage`
362 -- (See below for the actual view)
363 --
364 CREATE TABLE `staff_viewmessage` (
365 `message_idmessage` int(11)
366 ,`name` varchar(45)
367 ,`message` text
368 ,`timestamp` datetime
369 ,`fname` varchar(45)
370 ,`lname` varchar(45)
371 );
372
373 -- --------------------------------------------------------
374
375 --
376 -- Stand-in structure for view `staff_viewmessage2`
377 -- (See below for the actual view)
378 --
379 CREATE TABLE `staff_viewmessage2` (
380 `name` varchar(45)
381 ,`message` text
382 ,`staffid` int(11)
383 ,`timestamp` datetime
```

```
384 );
385
386 -- --------------------------------------------------------
387
388 --
389 -- Stand-in structure for view `staff_viewrequest`
390 -- (See below for the actual view)
391 --
392 CREATE TABLE `staff_viewrequest` (
393 `idrequest` int(11)
394 ,`staff_idstaff` int(11)
395 ,`user_iduser` int(11)
396 ,`status` enum('open','closed')
397 ,`timestamp` datetime
398 ,`replytimestamp` datetime
399 );
400
401 -- --------------------------------------------------------
402
403 --
404 -- Stand-in structure for view `staff_viewroom`
405 -- (See below for the actual view)
406 --
407 CREATE TABLE `staff_viewroom` (
408 `idroom` int(11)
409 ,`roomname` varchar(45)
410 ,`status` enum('open','closed')
411 );
412
413 -- --------------------------------------------------------
414
415 --
416 -- Stand-in structure for view `staff_viewservice`
417 -- (See below for the actual view)
418 --
419 CREATE TABLE `staff_viewservice` (
420 `idservice` int(11)
421 ,`name` varchar(45)
422 ,`price` decimal(6,2)
423 );
424
425 -- --------------------------------------------------------
426
427 --
428 -- Stand-in structure for view `staff_viewstaff`
429 -- (See below for the actual view)
430 --
431 CREATE TABLE `staff_viewstaff` (
432 `idstaff` int(11)
433 ,`salary` decimal(8,2)
434 ,`position` varchar(45)
435 ,`status` enum('active','nonactive')
436 ,`email` varchar(45)
```

```
,`fname` varchar(45)
,`lname` varchar(45)
,`address` varchar(100)
,`dob` date
,`personalid` varchar(15)
);

-- --------------------------------------------------------

--
-- Stand-in structure for view `staff_viewtransaction`
-- (See below for the actual view)
--
CREATE TABLE `staff_viewtransaction` (
`user_iduser` int(11)
,`fname` varchar(45)
,`lname` varchar(45)
,`amount` decimal(10,2)
,`method` enum('cash','creditcard')
,`type` enum('booking','service','','')
,`staff_idstaff` int(11)
,`timestamp` datetime
);

-- --------------------------------------------------------

--
-- Stand-in structure for view `staff_viewuser`
-- (See below for the actual view)
--
CREATE TABLE `staff_viewuser` (
`iduser` int(11)
,`usergroup_idusergroup` int(11)
,`password` varchar(45)
,`email` varchar(45)
,`fname` varchar(45)
,`lname` varchar(45)
,`address` varchar(100)
,`dob` date
,`personalid` varchar(15)
);

-- --------------------------------------------------------

--
-- Table structure for table `user`
--

CREATE TABLE `user` (
  `iduser` int(11) NOT NULL,
  `usergroup_idusergroup` int(11) NOT NULL,
  `password` varchar(45) NOT NULL,
  `email` varchar(45) NOT NULL,
```

```
  `fname` varchar(45) NOT NULL,
  `lname` varchar(45) NOT NULL,
  `address` varchar(100) DEFAULT NULL,
  `dob` date DEFAULT NULL,
  `personalid` varchar(15) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `user`
--

INSERT INTO `user` (`iduser`, `usergroup_idusergroup`, `password`, `email`, `fname`, `lname`, `address`, `dob`, `personalid`) VALUES
(1, 11, 'staff', 'staff', 'staff', 'staff', '123/36', '1996-11-01', '1150'),
(2, 1, 'user', 'user', 'user', 'user', '123/38', '1997-02-28', '1151'),
(3, 1, 'test', 'test', 'tt', 'tt', '23423', '2017-12-28', '11');

-- --------------------------------------------------------

--
-- Table structure for table `usergroup`
--

CREATE TABLE `usergroup` (
  `idusergroup` int(11) NOT NULL,
  `name` varchar(45) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `usergroup`
--

INSERT INTO `usergroup` (`idusergroup`, `name`) VALUES
(1, 'customer'),
(2, 'goldcustomer'),
(3, 'platinumcustomer'),
(11, 'staff');

-- --------------------------------------------------------

--
-- Table structure for table `usergroup_has_message`
--

CREATE TABLE `usergroup_has_message` (
  `usergroup_idusergroup` int(11) NOT NULL,
  `message_idmessage` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `usergroup_has_message`
--
```

```
541
542 INSERT INTO `usergroup_has_message` (`usergroup_idusergroup`, `
    message_idmessage`) VALUES
543 (1, 5),
544 (2, 2),
545 (3, 3);
546
547 -- --------------------------------------------------------
548
549 --
550 -- Structure for view `staff_viewbooking`
551 --
552 DROP TABLE IF EXISTS `staff_viewbooking`;
553
554 CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY
    DEFINER VIEW `staff_viewbooking`  AS  select `booking`.`idbooking` AS
    `idbooking`,`booking`.`user_iduser` AS `user_iduser`,`booking`.`
    roomnumber` AS `roomnumber`,`room`.`roomname` AS `roomname`,`booking
    `.`specialoffer_idspecialoffer` AS `discountcode`,`booking`.`
    staff_idstaff` AS `staff_idstaff`,`booking`.`payment_idpayment` AS `
    payment_idpayment`,`booking`.`fromdate` AS `fromdate`,`booking`.`
    todate` AS `todate`,`booking`.`status` AS `status`,`usergroup`.`name`
    AS `name`,`user`.`fname` AS `fname`,`user`.`lname` AS `lname` from
    (((`booking` left join `user` on((`user`.`iduser` = `booking`.`
    user_iduser`))) left join `room` on((`room`.`idroom` = `booking`.`
    room_idroom`))) left join `usergroup` on((`usergroup`.`idusergroup` =
    `user`.`usergroup_idusergroup`))) ;
555
556 -- --------------------------------------------------------
557
558 --
559 -- Structure for view `staff_viewmessage`
560 --
561 DROP TABLE IF EXISTS `staff_viewmessage`;
562
563 CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY
    DEFINER VIEW `staff_viewmessage`  AS  select `usergroup_has_message`.`
    message_idmessage` AS `message_idmessage`,`usergroup`.`name` AS `name
    `,`message`.`message` AS `message`,`message`.`timestamp` AS `timestamp
    `,`user`.`fname` AS `fname`,`user`.`lname` AS `lname` from ((((`
    usergroup_has_message` left join `message` on((`usergroup_has_message
    `.`message_idmessage` = `message`.`idmessage`))) left join `staff` on
    ((`staff`.`idstaff` = `message`.`staff_idstaff`))) left join `
    usergroup` on((`usergroup_has_message`.`usergroup_idusergroup` = `
    usergroup`.`idusergroup`))) left join `user` on((`user`.`iduser` = `
    staff`.`user_iduser`))) ;
564
565 -- --------------------------------------------------------
566
567 --
568 -- Structure for view `staff_viewmessage2`
569 --
570 DROP TABLE IF EXISTS `staff_viewmessage2`;
```

```
571
572 CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY
    DEFINER VIEW `staff_viewmessage2`  AS  select `usergroup`.`name` AS `
    name`,`message`.`message` AS `message`,`message`.`staff_idstaff` AS `
    staffid`,`message`.`timestamp` AS `timestamp` from ((`usergroup` join
    `usergroup_has_message`) join `message`) where ((`
    usergroup_has_message`.`usergroup_idusergroup` = `usergroup`.`
    idusergroup`) and (`usergroup_has_message`.`message_idmessage` = `
    message`.`idmessage`)) ;
573
574 -- --------------------------------------------------------
575
576 --
577 -- Structure for view `staff_viewrequest`
578 --
579 DROP TABLE IF EXISTS `staff_viewrequest`;
580
581 CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY
    DEFINER VIEW `staff_viewrequest`  AS  select `request`.`idrequest` AS
    `idrequest`,`request`.`staff_idstaff` AS `staff_idstaff`,`request`.`
    user_iduser` AS `user_iduser`,`request`.`status` AS `status`,`request
    `.`timestamp` AS `timestamp`,`request`.`replytimestamp` AS `
    replytimestamp` from `request` ;
582
583 -- --------------------------------------------------------
584
585 --
586 -- Structure for view `staff_viewroom`
587 --
588 DROP TABLE IF EXISTS `staff_viewroom`;
589
590 CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY
    DEFINER VIEW `staff_viewroom`  AS  select `room`.`idroom` AS `idroom
    `,`room`.`roomname` AS `roomname`,`room`.`status` AS `status` from `
    room` ;
591
592 -- --------------------------------------------------------
593
594 --
595 -- Structure for view `staff_viewservice`
596 --
597 DROP TABLE IF EXISTS `staff_viewservice`;
598
599 CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY
    DEFINER VIEW `staff_viewservice`  AS  select `service`.`idservice` AS
    `idservice`,`service`.`name` AS `name`,`service`.`price` AS `price`
    from `service` ;
600
601 -- --------------------------------------------------------
602
603 --
604 -- Structure for view `staff_viewstaff`
605 --
```

```
606 DROP TABLE IF EXISTS `staff_viewstaff`;
607
608 CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY
    DEFINER VIEW `staff_viewstaff`  AS  select `staff`.`idstaff` AS `
    idstaff`,`staff`.`salary` AS `salary`,`staff`.`position` AS `position
    `,`staff`.`status` AS `status`,`user`.`email` AS `email`,`user`.`fname
    ` AS `fname`,`user`.`lname` AS `lname`,`user`.`address` AS `address`,`
    user`.`dob` AS `dob`,`user`.`personalid` AS `personalid` from (`staff`
     left join `user` on((`user`.`iduser` = `staff`.`user_iduser`))) ;
609
610 -- --------------------------------------------------------
611
612 --
613 -- Structure for view `staff_viewtransaction`
614 --
615 DROP TABLE IF EXISTS `staff_viewtransaction`;
616
617 CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY
    DEFINER VIEW `staff_viewtransaction`  AS  select `payment`.`
    user_iduser` AS `user_iduser`,`user`.`fname` AS `fname`,`user`.`lname`
     AS `lname`,`payment`.`amount` AS `amount`,`payment`.`method` AS `
    method`,`payment`.`type` AS `type`,`payment`.`staff_idstaff` AS `
    staff_idstaff`,`payment`.`timestamp` AS `timestamp` from (`payment`
    left join `user` on((`user`.`iduser` = `payment`.`user_iduser`))) ;
618
619 -- --------------------------------------------------------
620
621 --
622 -- Structure for view `staff_viewuser`
623 --
624 DROP TABLE IF EXISTS `staff_viewuser`;
625
626 CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY
    DEFINER VIEW `staff_viewuser`  AS  select `user`.`iduser` AS `iduser
    `,`user`.`usergroup_idusergroup` AS `usergroup_idusergroup`,`user`.`
    password` AS `password`,`user`.`email` AS `email`,`user`.`fname` AS `
    fname`,`user`.`lname` AS `lname`,`user`.`address` AS `address`,`user
    `.`dob` AS `dob`,`user`.`personalid` AS `personalid` from `user` ;
627
628 --
629 -- Indexes for dumped tables
630 --
631
632 --
633 -- Indexes for table `booking`
634 --
635 ALTER TABLE `booking`
636   ADD PRIMARY KEY (`idbooking`),
637   ADD KEY `fk_booking_user1_idx` (`user_iduser`),
638   ADD KEY `fk_booking_room1_idx` (`room_idroom`),
639   ADD KEY `fk_booking_specialoffer1_idx` (`specialoffer_idspecialoffer`),
640   ADD KEY `fk_booking_staff1_idx` (`staff_idstaff`),
641   ADD KEY `fk_booking_payment1_idx` (`payment_idpayment`);
```

```
642
643 --
644 -- Indexes for table `message`
645 --
646 ALTER TABLE `message`
647   ADD PRIMARY KEY (`idmessage`),
648   ADD KEY `fk_message_staff1_idx` (`staff_idstaff`);
649
650 --
651 -- Indexes for table `payment`
652 --
653 ALTER TABLE `payment`
654   ADD PRIMARY KEY (`idpayment`),
655   ADD KEY `fk_payment_user1_idx` (`user_iduser`),
656   ADD KEY `fk_payment_staff1_idx` (`staff_idstaff`);
657
658 --
659 -- Indexes for table `request`
660 --
661 ALTER TABLE `request`
662   ADD PRIMARY KEY (`idrequest`),
663   ADD KEY `fk_request_user1_idx` (`user_iduser`),
664   ADD KEY `fk_request_staff1_idx` (`staff_idstaff`);
665
666 --
667 -- Indexes for table `room`
668 --
669 ALTER TABLE `room`
670   ADD PRIMARY KEY (`idroom`),
671   ADD KEY `fk_room_roomtype1_idx` (`roomtype_idroomtype`);
672
673 --
674 -- Indexes for table `roomtype`
675 --
676 ALTER TABLE `roomtype`
677   ADD PRIMARY KEY (`idroomtype`);
678
679 --
680 -- Indexes for table `service`
681 --
682 ALTER TABLE `service`
683   ADD PRIMARY KEY (`idservice`);
684
685 --
686 -- Indexes for table `specialoffer`
687 --
688 ALTER TABLE `specialoffer`
689   ADD PRIMARY KEY (`idspecialoffer`),
690   ADD KEY `fk_specialoffer_staff1_idx` (`staff_idstaff`);
691
692 --
693 -- Indexes for table `staff`
694 --
```

```
ALTER TABLE `staff`
  ADD PRIMARY KEY (`idstaff`),
  ADD KEY `fk_staff_user1_idx` (`user_iduser`);

--
-- Indexes for table `user`
--
ALTER TABLE `user`
  ADD PRIMARY KEY (`iduser`),
  ADD KEY `fk_user_usergroup1_idx` (`usergroup_idusergroup`);

--
-- Indexes for table `usergroup`
--
ALTER TABLE `usergroup`
  ADD PRIMARY KEY (`idusergroup`);

--
-- Indexes for table `usergroup_has_message`
--
ALTER TABLE `usergroup_has_message`
  ADD PRIMARY KEY (`usergroup_idusergroup`,`message_idmessage`),
  ADD KEY `fk_usergroup_has_message_message1_idx` (`message_idmessage`),
  ADD KEY `fk_usergroup_has_message_usergroup1_idx` (`
    usergroup_idusergroup`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `booking`
--
ALTER TABLE `booking`
  MODIFY `idbooking` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;

--
-- AUTO_INCREMENT for table `message`
--
ALTER TABLE `message`
  MODIFY `idmessage` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;

--
-- AUTO_INCREMENT for table `payment`
--
ALTER TABLE `payment`
  MODIFY `idpayment` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;

--
-- AUTO_INCREMENT for table `request`
--
ALTER TABLE `request`
  MODIFY `idrequest` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
```

```
747
748 --
749 -- AUTO_INCREMENT for table `room`
750 --
751 ALTER TABLE `room`
752   MODIFY `idroom` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=8;
753
754 --
755 -- AUTO_INCREMENT for table `roomtype`
756 --
757 ALTER TABLE `roomtype`
758   MODIFY `idroomtype` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
759
760 --
761 -- AUTO_INCREMENT for table `service`
762 --
763 ALTER TABLE `service`
764   MODIFY `idservice` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;
765
766 --
767 -- AUTO_INCREMENT for table `specialoffer`
768 --
769 ALTER TABLE `specialoffer`
770   MODIFY `idspecialoffer` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT
      =3;
771
772 --
773 -- AUTO_INCREMENT for table `staff`
774 --
775 ALTER TABLE `staff`
776   MODIFY `idstaff` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;
777
778 --
779 -- AUTO_INCREMENT for table `user`
780 --
781 ALTER TABLE `user`
782   MODIFY `iduser` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
783
784 --
785 -- AUTO_INCREMENT for table `usergroup`
786 --
787 ALTER TABLE `usergroup`
788   MODIFY `idusergroup` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT
      =12;
789
790 --
791 -- Constraints for dumped tables
792 --
793
794 --
795 -- Constraints for table `booking`
796 --
797 ALTER TABLE `booking`
```

```
798   ADD CONSTRAINT `fk_booking_payment1` FOREIGN KEY (`payment_idpayment`)
        REFERENCES `payment` (`idpayment`) ON DELETE CASCADE ON UPDATE NO
        ACTION,
799   ADD CONSTRAINT `fk_booking_room1` FOREIGN KEY (`room_idroom`)
        REFERENCES `room` (`idroom`) ON DELETE CASCADE ON UPDATE NO ACTION,
800   ADD CONSTRAINT `fk_booking_specialoffer1` FOREIGN KEY (`
        specialoffer_idspecialoffer`) REFERENCES `specialoffer` (`
        idspecialoffer`) ON DELETE CASCADE ON UPDATE NO ACTION,
801   ADD CONSTRAINT `fk_booking_staff1` FOREIGN KEY (`staff_idstaff`)
        REFERENCES `staff` (`idstaff`) ON DELETE CASCADE ON UPDATE NO ACTION,
802   ADD CONSTRAINT `fk_booking_user1` FOREIGN KEY (`user_iduser`)
        REFERENCES `user` (`iduser`) ON DELETE CASCADE ON UPDATE NO ACTION;
803
804 --
805 -- Constraints for table `message`
806 --
807 ALTER TABLE `message`
808   ADD CONSTRAINT `fk_message_staff1` FOREIGN KEY (`staff_idstaff`)
        REFERENCES `staff` (`idstaff`) ON DELETE NO ACTION ON UPDATE NO ACTION
        ;
809
810 --
811 -- Constraints for table `payment`
812 --
813 ALTER TABLE `payment`
814   ADD CONSTRAINT `fk_payment_staff1` FOREIGN KEY (`staff_idstaff`)
        REFERENCES `staff` (`idstaff`) ON DELETE CASCADE ON UPDATE NO ACTION,
815   ADD CONSTRAINT `fk_payment_user1` FOREIGN KEY (`user_iduser`)
        REFERENCES `user` (`iduser`) ON DELETE CASCADE ON UPDATE NO ACTION;
816
817 --
818 -- Constraints for table `request`
819 --
820 ALTER TABLE `request`
821   ADD CONSTRAINT `fk_request_staff1` FOREIGN KEY (`staff_idstaff`)
        REFERENCES `staff` (`idstaff`) ON DELETE NO ACTION ON UPDATE NO ACTION
        ,
822   ADD CONSTRAINT `fk_request_user1` FOREIGN KEY (`user_iduser`)
        REFERENCES `user` (`iduser`) ON DELETE NO ACTION ON UPDATE NO ACTION;
823
824 --
825 -- Constraints for table `room`
826 --
827 ALTER TABLE `room`
828   ADD CONSTRAINT `fk_room_roomtype1` FOREIGN KEY (`roomtype_idroomtype`)
        REFERENCES `roomtype` (`idroomtype`) ON DELETE NO ACTION ON UPDATE NO
        ACTION;
829
830 --
831 -- Constraints for table `specialoffer`
832 --
833 ALTER TABLE `specialoffer`
834   ADD CONSTRAINT `fk_specialoffer_staff1` FOREIGN KEY (`staff_idstaff`)
```

```
    REFERENCES `staff` (`idstaff`) ON DELETE NO ACTION ON UPDATE NO ACTION
    ;

--
-- Constraints for table `staff`
--
ALTER TABLE `staff`
  ADD CONSTRAINT `fk_staff_user1` FOREIGN KEY (`user_iduser`) REFERENCES
    `user` (`iduser`) ON DELETE NO ACTION ON UPDATE NO ACTION;

--
-- Constraints for table `user`
--
ALTER TABLE `user`
  ADD CONSTRAINT `fk_user_usergroup1` FOREIGN KEY (`usergroup_idusergroup
    `) REFERENCES `usergroup` (`idusergroup`) ON DELETE NO ACTION ON
    UPDATE NO ACTION;

--
-- Constraints for table `usergroup_has_message`
--
ALTER TABLE `usergroup_has_message`
  ADD CONSTRAINT `fk_usergroup_has_message_message1` FOREIGN KEY (`
    message_idmessage`) REFERENCES `message` (`idmessage`) ON DELETE NO
    ACTION ON UPDATE NO ACTION,
  ADD CONSTRAINT `fk_usergroup_has_message_usergroup1` FOREIGN KEY (`
    usergroup_idusergroup`) REFERENCES `usergroup` (`idusergroup`) ON
    DELETE NO ACTION ON UPDATE NO ACTION;
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

A.2 Lines count

```

Line counts for project /Users/macbookair13/Documents/GitHub/DatabaseLab.
Generated by the Atom editor package Line-Count on December 14 2017
    19:02.
Counts are in order of source, comments, and total.

Files
-----
     5    0     5  database/addarray.php
    40    0    51  database/booking.php
    76    0    85  database/bookinglogin.php
     8    0     8  database/connect.php
    26    0    34  database/customer.php
   397   31   454  database/default.css
     5    0     5  database/destroy.php
   109    0   113  database/facility.php
    81    0    91  database/helperfunctions.php
```

```
    83    0     94  database/home.php
     1    0      1  database/img/fb.svg
    75    0     84  database/login.php
    28    0     28  database/loginaction.php
     5    0      5  database/logout.php
    31    0     38  database/mainstaff.php
    28    0     35  database/mybooking.php
    42    0     50  database/myprofile.php
    39    0     47  database/mytransaction.php
   139    7   162  database/newbooking.php
    46    2     53  database/notice.php
    37    0     39  database/php/addroom.php
    53    0     53  database/php/addroomtype.php
    10    0     10  database/php/addroomtype_action.php
     8    0      8  database/php/connect.php
    35    2     40  database/php/customerdashboard.php
     0    0      1  database/php/editroom.php
     2    0      3  database/php/formaction.php
    50    2     54  database/php/login.php
    50    0     51  database/php/register.php
    61    0     67  database/php/room.php
     4    0      6  database/php/sessionexample.php
    35    2     39  database/php/staffdashboard.php
    14    0     15  database/registeraction.php
    27    0     36  database/request.php
    26    0     34  database/review.php
    79    0     83  database/room.php
    78    1     89  database/selectguest.php
    44    0     53  database/sendrequest.php
    38    2     42  database/temp.html
    97    0   106  database/temp.php
    37    0     46  database/template.php
     5    0      5  database/unset.php
    33    0     42  database/viewrequest.php
    14    0     14  newpage/addcart.php
    17    6     24  newpage/applycoupon.php
    64    1     79  newpage/booking.php
    31    1     38  newpage/booking_staff.php
    43    1     50  newpage/booking_staff_checkin.php
    62    1     76  newpage/booking_staff_create.php
    62    1     79  newpage/booking_staff_edit.php
    78    1     90  newpage/booking_staff_view.php
    18    0     19  newpage/calprice.php
    21    0     21  newpage/checkdateAction.php
    21    0     21  newpage/checkin_action.php
   173    2   187  newpage/confirm.php
     8    0      8  newpage/connect.php
    85    4     98  newpage/createaction.php
    41   14     61  newpage/createbookingpayment.php
    50    2     60  newpage/dashboard.php
    38    1     47  newpage/dashboard_staff.php
  1195   75  1374  newpage/default.css
     6    0      6  newpage/del_servicecart.php
```

```
  18   0   19  newpage/deleteaction.php
  69   1   81  newpage/editaction.php
  41   0   50  newpage/editprofile.php
  28   1   30  newpage/editprofileAction.php
  91   1   98  newpage/facility.php
  91   1  105  newpage/finishbooking.php
  28   1   36  newpage/fullpage_template.php
 153  10  180  newpage/helperfunction.php
  81   1   94  newpage/home.php
   1   0    1  newpage/img/fb.svg
 105   1  113  newpage/login.php
  41   4   46  newpage/loginaction.php
   5   0    5  newpage/logout.php
  30   1   37  newpage/message_staff.php
  55   1   68  newpage/message_staff_edit.php
  68   1   83  newpage/message_staff_sent.php
  76   1   88  newpage/message_staff_view.php
  44   0   54  newpage/myprofile.php
  49   0   61  newpage/mytransaction.php
  27   1   35  newpage/newbooking.php
  26   0   26  newpage/registeraction.php
   9   0    9  newpage/removecart.php
   4   1    5  newpage/replymessage.php
  28   0   37  newpage/request.php
  29   1   37  newpage/request_staff.php
  76   1   93  newpage/request_staff_edit.php
  75   1   86  newpage/request_staff_view.php
  33   1   39  newpage/room.php
  29   1   36  newpage/room_staff.php
  51   1   61  newpage/room_staff_edit.php
  64   3   78  newpage/room_staff_view.php
 120  10  149  newpage/selectroom.php
  45   1   57  newpage/sendrequest.php
  22   2   24  newpage/sendrequestAction.php
  31   1   39  newpage/service_staff.php
 105   2  118  newpage/service_staff_charge.php
  47   1   64  newpage/service_staff_create.php
  47   2   63  newpage/service_staff_edit.php
  78   1   91  newpage/service_staff_view.php
  30   1   37  newpage/staff_staff.php
  54   1   69  newpage/staff_staff_create.php
  61   1   76  newpage/staff_staff_edit.php
  76   1   86  newpage/staff_staff_view.php
  59   1   70  newpage/success.php
  29   1   37  newpage/transaction_staff.php
  64   1   75  newpage/transaction_staff_view.php
  30   1   37  newpage/user_staff.php
  52   1   67  newpage/user_staff_create.php
  80   1   97  newpage/user_staff_edit.php
  78   1   90  newpage/user_staff_view.php
  32   2   34  newpage/viewreqmessage.php
  63   1   74  newpage/viewrequest.php
  47  15   62  newpage/walkinbooking.php
```

```
123
124 Directories
125 -----------
126   2087   49  2365  database
127      1    0     1  database/img
128    345    6   371  database/php
129   4802  195  5659  newpage
130      1    0     1  newpage/img
131
132 Types
133 -----
134   1592  106  1828  css
135     38    2    42  html
136   5257  136  6152  php
137      2    0     2  svg
138
139 Total
140 -----
141   6889  244  8024
```

A.3 Github URL

This is the url of our project's repository `https://github.com/naughtybunnies/DatabaseLab/tree/aftermid`