INDIRA GANDHI NATIONAL OPEN UNIVERSITY

MCSP-060

Hotel Management System

By

Mandeep Kaur

Enrolment No: 146438712

Under Guidance

Of

Mr. Prabal Kumar Joshi

Submitted to the School of Computer and Information Science, IGNOU

In partial fulfillment of the requirements

For the award of the degree

Masters of Computer applications (MCA)

INDIRA GANDHI NATIONAL OPEN UNIVERSITY

Maidan Ghari

New Delhi-110068

Index

Introduction/Objectives

Project title “MANN HOTEL” (a project for keeping customers record and also calculate customer bill slip and managers salary) The name of project is “MANN HOTEL”. The objective of the project is to computerize the system of the hotel. “MANN HOTEL” is the project not only keeps the record of various people like customers, manager etc. but as well as it reduce the extensive paper work in the present system. It wills make the system more versatile and user friendly. It also calculates the proper billing slip of high level and middle level customers. This Project is a fine thought to make the complex procedure of the Hotel management system to an easy manner which is systematic, modular designed, selective menu based user display. The modular design and constructed system is very much user oriented in which user can easily understand the tools and can do edit of his own choice. The system is not any tough more and does not possesses many applications but it is made by focusing on the maintaining records employee’s actions in a computerized system rather than time taking and cumbersome manual system.

The project is a software application that can be easily handled by minimum educated and simple computer knowledge person without any option of error. Two kinds of users can handle the system.

1. Online Users,
2. Administrator or Hotel Management.

The Online users are the customers or the staff who can see the news and updates of the Hotel and the Administrator are responsible for updating the Hotel details on computer. The Administrator is the authorized user who has power to change or edit the updates as well as the Password. In case of the forgetting of password there is provision to password recovery and Logout and Login in the system.

The word hotel is derived from the French hotel (coming from the same origin as hospital), which referred to a French version of a building seeing frequent visitors, and providing care, rather than a place offering accommodation. In contemporary French usage, hotel now has the same meaning as the English term, and hotel particulier is used for the old meaning, as well as "hotel" in some place names such as Hotel-Dieu (in Paris), which has been a hospital since the Middle Ages. The French spelling, with the circumflex, was also used in English, but is now rare. The circumflex replaces the 's' found in the earlier hostel spelling, which over time took on a new, but closely related meaning. Grammatically, hotels usually take the definite article – hence "The Astoria Hotel" or simply "The Astoria."

Objective

Internet and Web technologies are growing day by day and today everyone is aware of this fact. No one has remained untouched from this invention and now our day is not complete without browsing through our favourite website whether that is *Social Media, Online shopping or Entertainment*. The **Hotel Management System** we are creating adheres to this fact and focuses on providing its end-users a one-stop portal for organising and managing multiple bookings at the same time. Also it helps to keep track of all the activities a booking goes through during its life cycle.

During the past several decades personnel function has been transformed from a relatively obscure record keeping staff to central and top level management. They are

many factors that have influenced this transformation like technological advances, professionalism and general recognition of human beings as most important to resources.

- A computer based management system is designed to handle all the primary information required to calculate the monthly statements.
- Maintained to handle all the details required for the correct statement calculation and generation.
- This project intends to introduce more user friendliness in the various activities such as record updating, maintenance and searching.
- The searching of records has been made quite simple as all the details of the customer can be obtained by simply keying in the identification of that customer.
- Similarly, record maintenance and updating can also be accomplished by using the identification of the customer with all the details being automatically generated. These details are also being promptly automatically updating in the master file thus keeping the record absolutely up-to-date.
- The entire information has maintained in the database or files and whoever wants to retrieve can't retrieve, only authorization user can retrieve the necessary information which can be easily be accessible from the file.

The main objective of the entire activity is to automate the processes of day to day activity of Hotel like:

- Room activities
- Admission of new customer
- Assign a room according to customer demands
- Checkout time and releasing the room
- Finally compute the bill
- Packages available
- Advance online booking
- List of regular customer
- Email facility
- Feedback

Achievements

Hotel Management system has been developed with in a way that any number of persons can use the portal and start management of their own personal bookings in no time.

Also on the same time the hotel managers can keep track of all the bookings and their orders as well.

Making the system idel for small scaled hotels as well by helping them keep track of every guest in their hotel from one unified portal.

System Analysis

Every software project goes through a certain set of stages and processes that are standard for almost every company that has worked in the IT Industry. These stages are referred to as the SDLC or Software Development Life Cycle of the project. There are a lot of SDLC Models available to us for the process of software engineering. Some are quite simple and others are complicated enough to take months and months of practice to master them. For the current project, we have adopted the **'Spiral Model'** of software engineering.

Identification of Need

Preliminary Investigation

Feasibility Study

Creating any piece of software is a task that requires a lot of dedication and timeliness and creating an entire system takes more time than a basic software. It requires future planning before we even begin to start writing the first line of code. The very first task that we are required to perform is to conduct a feasibility study.

According to Wikipedia, **Feasibility Study** is an assessment of the practicality of a proposed project or system.

So we begin by finding if such a software can actually be used here under four categories.

Technical Feasibility

Determines whether the proposed system conflicts with legal requirements, e.g. a data processing system must comply with the local data protection regulations and if the proposed venture is acceptable in accordance to the laws of the land.

Our proposed System (website) runs on PHP stack and hence is open source will be easy to develop and launch as well. Which makes our system technically feasible.

Operational Feasibility

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters as reliability, maintainability, supportability, usability, producibility, disposability, sustainability, affordability and others. These parameters are required to be considered at the early stages of design if desired operational behaviours are to be realised. A system design and development requires appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters. A system may serve its intended purpose most effectively when its technical and operating characteristics are engineered into the design. Therefore, operational feasibility is a critical aspect of systems engineering that needs to be an integral part of the early design phases.

As discussed earlier our website will allow the user to perform all the basic event management tasks making our website operationally feasible.

Schedule Feasibility

A project will fail if it takes too long to be completed before it is useful. Typically this means estimating how long the system will take to develop, and if it can be completed in a given time period using some methods like payback period. Schedule feasibility is a measure of how reasonable the project timetable is. Given our technical expertise, are the project deadlines reasonable? Some projects are initiated with specific deadlines. It is necessary to determine whether the deadlines are mandatory or desirable.

The project of this website is required as the partial fulfilment of the MCA it needs to be on schedule and be submitted before 30th April 2017.

As per our calculations, this project is feasible in terms of schedule.

Project Planning

When you think of building a website, your thoughts rotate around two main issues – price and time. These two values depend largely on the size and scope of the project. To outline the whole development process, you can create a website development timeline, adding tasks and establishing milestones for your project. It is the best way to track your project implementation to make sure you keep up with the deadline.

We've prepared detailed description of the whole website development process, estimated time for each step and a checklist to double check you don't miss anything.

Despite conventional wisdom, the core part of website development and design is not necessary for the coding process. Indeed, such technologies as HTML, CSS, and JavaScript give the web we know its shape and define the way we interact with the information. But usually stay behind the scenes and, at the same time, remain the crucial part of website development life cycle are the stages of preliminary information gathering, detailed planning, and post-launch maintenance.

Spiral Model

The spiral model, originally proposed by Boehm, is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model. The spiral model is similar to the incremental model, with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spirals builds on the baseline spiral. It's one of the software development models like Waterfall, Agile, V-Model. A spiral model is divided into a number of framework activities, also called task regions. Typically, there are between three and six task regions. In below figure depicts a spiral model that contains six task regions:

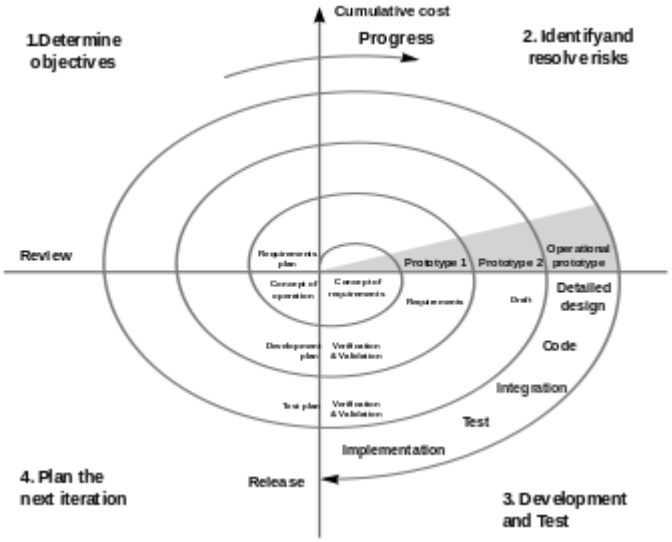


Figure 1 Spiral Model

Customer communication
—tasks required to establish effective communication between developer and customer.

Planning
—tasks required to define resources, timelines, and other project related information.

Risk analysis
—tasks required to assess both technical and management risks.

Engineering
—tasks required to build one or more representations of the application.

Construction and release
—tasks required to construct, test, install, and provide user support (e.g., documentation and training).

Customer evaluation
—tasks required to obtain customer feedback based on evaluation of the software representations that were created during the engineering stage and implemented during the installation stage.

Application of Spiral Model
The Spiral Model is widely used in the software industry as it is in sync with the natural development process of any product, i.e. learning with maturity which involves minimum risk for the customer as well as the development firms.

The following pointers explain the typical uses of a Spiral Model:

- When there is a budget constraint and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customer is not sure of their requirements which is usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

Advantages of Spiral model:

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date.
- Software is produced early in the software life cycle.

Disadvantages of Spiral model

- Can be a costly model to use.
- Risk analysis requires highly specific expertise.
- Project’s success is highly dependent on the risk analysis phase.
- Doesn’t work well for smaller projects.

When to use Spiral model

- When costs and risk evaluation is important
- For medium to high-risk projects

- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex
- New product line
- Significant changes are expected (research and exploration)

Conclusion

Each spiral can be termed as a loop and each loop is a separate development process in a spiral model. The four activities (Planning, Risk analysis, engineering and evaluation) form the intermediary phases of a spiral model and is repeated again for each loop.

This model is very good to use for larger projects where you can develop and deliver smaller prototypes and can enhance it to make the larger software. The implementation of this model requires experienced resources as risk analysis is a very integral part of this model and risk analysis requires expertise and as a result this model becomes costly.

We can draw four conclusions from the data presented:

a. The risk-driven nature of the spiral model is more adaptable to the full range of software project situations than are the primarily document-driven approaches such as the waterfall model or the primarily code-driven approaches such as evolutionary development. It is particularly applicable to very large, complex, ambitious soft-wane systems.
b. The spiral model has been quite successful in its largest application to date: the development and enhancement of the TRW-SPS. Overall, it achieved a high level of software support environment capability in a very short time and provided the flexibility necessary to accommodate a high dynamic range of technical alternatives and user objectives.
c. The spiral model is not yet as fully elaborated as the more established models. Therefore, the spinal model can be applied by experienced personnel, but it needs further elaboration in such areas as contracting, specifications, milestones, reviews, scheduling, status monitoring, and risk area identification to be fully usable in all situations.

Partial implementations of the spiral model, such as the risk management plan, are compatible with most current process models and are very helpful in overcoming major sources of project risk.

In this article, we'll take a look at how the general website development process may look like. The overall number of development stages usually varies from five to eight, but every time the whole picture stays pretty much the same. Let's choose the average value. So, here are seven main steps:

i. Information Gathering
ii. Planning
iii. Design
iv. Content Writing and Assembly
v. Coding
vi. Testing
vii. Review
viii. Launch
ix. Maintenance

Step 1: Gathering Information

Purpose, Main Goals, and Target Audience

This stage, the stage of discovering and researching, determines how the subsequent steps will look like. The most important task at this point is to get the clear understanding of your future website purposes, the main goals you wish to get, and the target audience you want to attract to your site. Such kind of a website development questionnaire helps to develop the best strategy for further project management.

News portal differs from the entertainment websites, and online resources for teenagers looks different than sites for adults. Different types of websites provide visitors with different functionality which means that different technologies should be used according to the purposes. A well described and detailed plan made on the basis of this pre-development data can protect you from spending extra resources on solving the unexpected issues such as design changing or adding the functionality that wasn't initially planned.

Step 2: Planning

Sitemap and Wireframe Creation

At this stage of website development cycle, the developer creates the data that can give to a customer an opportunity to judge how the entire site will look like.

On the basis of the information that was gathered together in the previous phase, the sitemap is created. Here is the sitemap of our **Hotel Management** website:

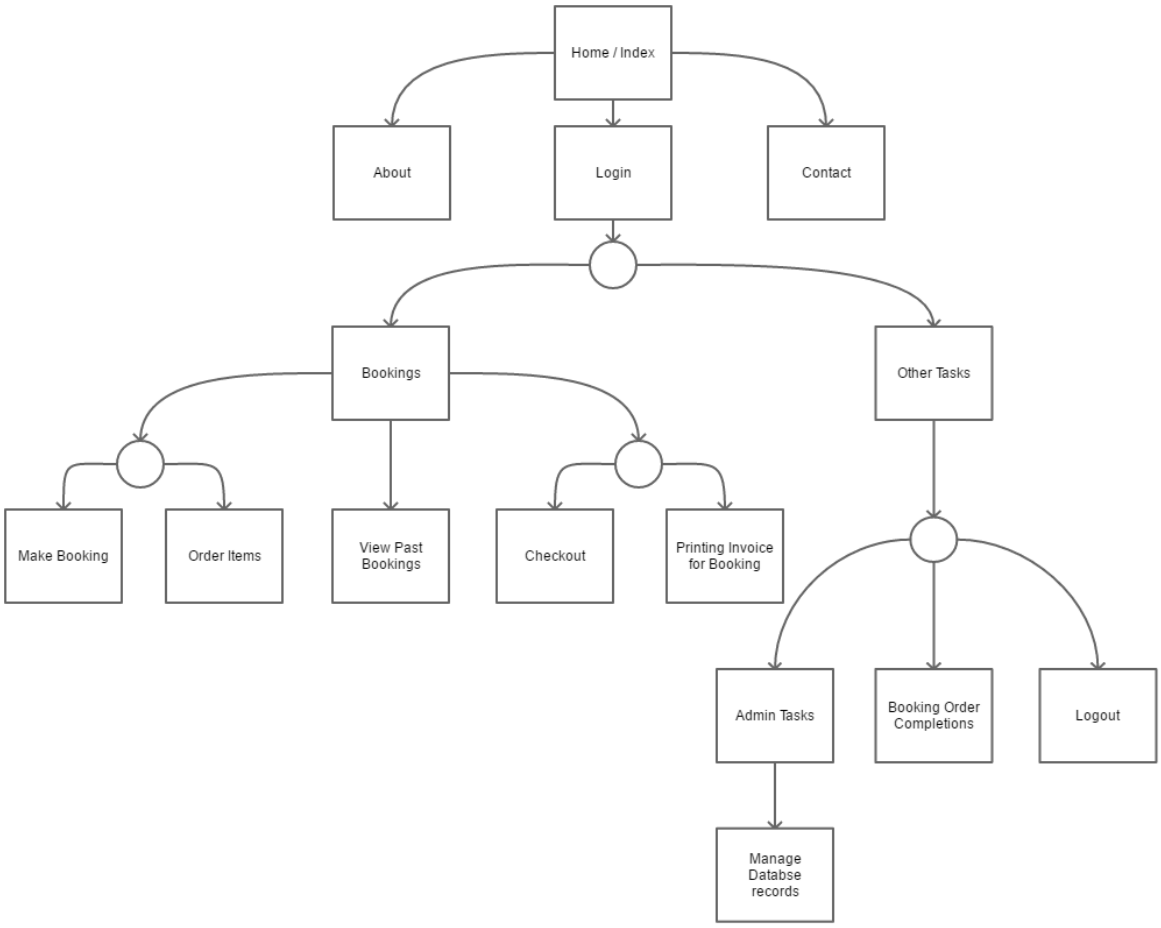


Figure 2 Site Map

The sitemap should describe the relations between the main areas of a website. Such representation could help understand how usable the final product will be. It can show you the *"relationship"* between the different pages of a website, so you can judge how easy it will be for the end-user to find the required information or service if he starts from the main page. The main reason behind the sitemap creation is to build a user-friendly and easy to navigate website.

The sitemap allows you to understand how the inner structure of a website looks like, but doesn't describe the user interface. Sometimes, before you start to code or even work on a design, there's a necessity to get approval from a customer that everything looks fine so you can begin the next phase of development. In this case, a ***wireframe or mock-up*** is created. A *wireframe* is a visual representation of user interface that you're going to create. But it doesn't contain any design elements such as colors, logos, etc. It only describes the elements that will be added to the page and their location. It's artless and cheap in production sketch.

You can use any mockup for this purpose. We used Pencil Project. Here's how the wireframe can look like:

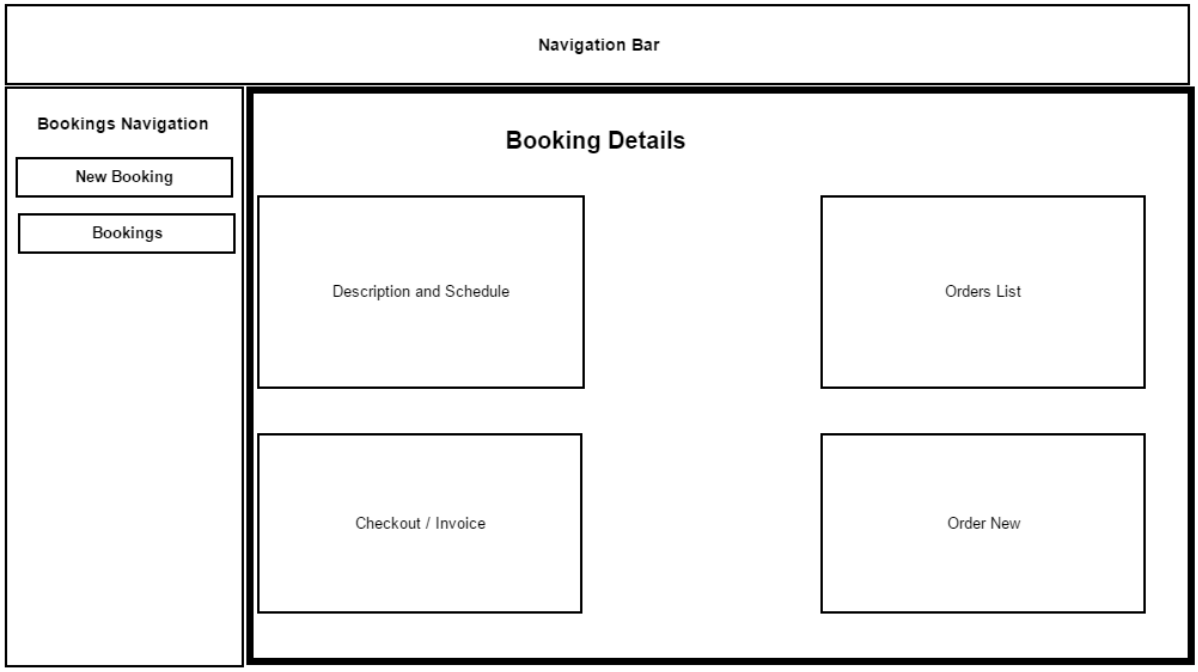


Figure 3 Wireframe

The other important thing is to select technology stack – programming language, frameworks, CMS that you're going to use.

Step 3: Design

Page Layouts, Review, and Approval Cycle

During the design phase, your website takes shape. All the visual content, such as images, photos, and videos is created at this step. Once again, all the info that was gathered through the first phase is crucial. The customer and target audience must be kept in mind while you work on a design.

Website layout is the result of designer's work. It can be a graphic sketch or an actual graphic design. The primary function of the layout is to represent the information structure, visualize the content, and demonstrate the basic functional. Layouts contain colors, logos, images and can give a general understanding of the future product.

After that, the customer can review the layout and send you his feedback. If the client is not sure about some aspects of your design, you should change the layout and send it back to him. This cycle should be repeated until the customer is completely satisfied.

Step 4: Content Writing and Assembly

Content writing and compiling usually overlaps with other stages of website creation, and its role can't be underestimated. At this step it is necessary to put in writing the very essence you'd like to communicate to the audience of your website and add calls-to-action. Content writing involves also creation of catching headlines, text editing, writing new text, compiling the existing text, etc., which takes time and effort. As a rule, the client undertakes to provide website content ready to migrate to the site. It is better when all website content is provided before or during website coding.

Step 5: Coding

At this step, you can finally start creating the website itself. Graphic elements that have been designed during the previous stages should be used to create an actual website. Usually, the home page is created first, and then all sub-pages are added, according to the website hierarchy that was previously created in the form of a sitemap. Frameworks and CMS should be implemented to make sure that server can handle the installation and set-up smoothly.

All static web page elements that were designed during the mock-up and layout creation should be created and tested. Then, special features and interactivity should be added. A deep understanding of every website development technology that you're going to use is crucial at this phase.

When you use CMS for site creation, you can also install CMS plugins at this step if there's a need. The other important step is SEO (Search Engine Optimization). SEO is the optimization of website elements (e.g., title, description, and keyword) that can help your site achieve higher rankings in the search engines. And, once again, valid code is pretty important for SEO.

Step 6: Testing, Review and Launch

Testing is probably the most routine part of a process. Every single link should be tested to make sure that there are no broken bones among them. You should check every form, every script, run a spell-checking software to find possible typos. Use code validators to check if your code follows the current web standards. Valid code is necessary, for example, if cross-browser compatibility is important for you.

After you check and re-check your website, it's time to upload it to a server. An FTP (File Transfer Protocol) software is used for that purpose. After you deployed the files, you should run yet another, final test to be sure that all your files have been installed correctly.

Step 7: Maintenance

Opinion Monitoring and Regular Updating

What's important to remember is that a website is more a service than a product. It's not enough to "deliver" a website to a user. You should also make sure that everything

works fine, and everybody is satisfied and always be prepared to make changes in another case.

Feedback system added to the site will allow you to detect possible problems the end-users face. The highest priority task in this case is to fix the problem as fast as you can. If you won't, you may find one day that your users prefer to use another website rather than put up with the inconvenience.

The other important thing is keeping your website up to date. If you use a CMS, regular updates will prevent you from bugs and decrease security risks.

Conclusion

You should always keep in mind that website development project doesn't start with coding and doesn't end after the day you finally launch your website. The phase of preparation affects all subsequent stages, defining how productive the development process will be. A profound and deep discovery of such aspects like age, sex, and interests of your end-user may become the key to success. The post-launch period is rather significant. Your project should be agile and flexible enough to have a possibility to change your website according to users' feedback or spirit of the time. Keeping in mind that there's no such thing as insignificant website development phase will prevent you from unexpected troubles and give you confidence that everything flows as it should, and you have full control over the project.

Project Scheduling

The project schedule is the tool that communicates what work needs to be performed, which resources of the organization will perform the work and the timeframes in which that work needs to be performed. The project schedule should reflect all of the work associated with delivering the project on time. Without a full and complete schedule, the project manager will be unable to communicate the complete effort, in terms of cost and resources, necessary to deliver the project.

Online project management software allows project managers to track project schedules, resources, budgets and project related assets in real time. The project schedule can be viewed and updated by team members associated with the project, keeping everyone well informed on the overall project status.

Methods

Before a project schedule can be created, the schedule maker should have a work breakdown structure (WBS), an effort estimate for each task, and a resource list with availability for-each resource. If these components for the schedule are not available, they can be created with a consensus-driven estimation method like Wideband Delphi. The reason for this is that a schedule itself is an estimate: each date in the schedule is estimated, and if those dates do not have the buy-in of the people who are going to do the work, the schedule will be inaccurate.

In order for a project schedule to be healthy, the following criteria must be met

The schedule must be constantly (weekly works best) updated.

The EAC (Estimation at Completion) value must be equal to the baseline value.

The remaining effort must be appropriately distributed among team members (taking vacations into consideration).

The schedule structure may closely follow and include citations to the index of work breakdown structure or deliverables, using decomposition or templates to describe the activities needed to produce the deliverables defined in the WBS.

A schedule may be assessed for the quality of the schedule development and the quality of the schedule management.

Work Breakdown Structure (WBS)

The building blocks of a schedule start with a Work Breakdown Structure (WBS). The WBS is a hierarchical reflection of all the work in the project in terms of deliverables. In order to produce these deliverables, work must be performed.

A typical approach in developing a WBS is to start at the highest level, with the product of the project. For example, you are assigned as the project manager of a New Product Development project. The new product you are developing is a new toy for children age's five through nine. The objective of this product development project is to increase the revenue of the organization by ten percent.

WBS is a hierarchical and incremental decomposition of the project into phases, deliverables and work packages. It is a tree structure, which shows a subdivision of effort required to achieve an objective; for example a program, project, and contract. In a project or contract, the WBS is developed by starting with the end objective and successively subdividing it into manageable components in terms of size, duration, and responsibility (e.g., systems, subsystems, components, tasks, subtasks, and work packages) which include all steps necessary to achieve the objective.

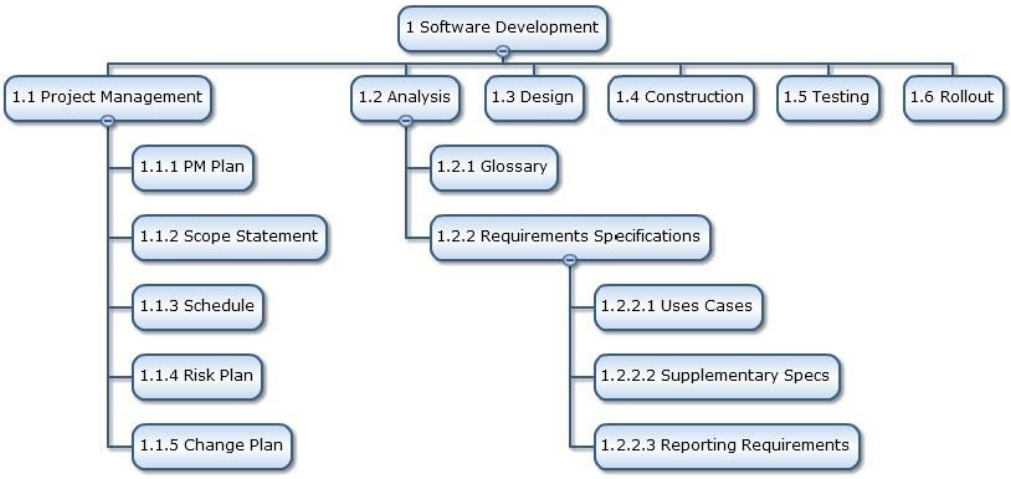


Figure 4 WBS

The work breakdown structure provides a common framework for the natural development of the overall planning and control of a contract and is the basis for dividing work into definable increments from which the statement of work can be developed and technical, schedule, cost, and labour hour reporting can be established.

A work breakdown structure permits summing of subordinate costs for tasks, materials, etc., into their successively higher level "parent" tasks, materials, etc. For each element of the work breakdown structure, a description of the task to be performed is generated. This technique (sometimes called a system breakdown structure) is used to define and organize the total scope of a project.

The WBS is organized around the primary products of the project (or planned outcomes) instead of the work needed to produce the products (planned actions). Since the planned outcomes are the desired ends of the project, they form a relatively stable set of categories in which the costs of the planned actions needed to achieve them can be collected. A well-designed WBS makes it easy to assign each project activity to one and

only one terminal element of the WBS. In addition to its function in cost accounting, the WBS also helps map requirements from one level of system specification to another, for example a requirements cross reference matrix mapping functional requirements to high level or low level design documents. The WBS may be displayed horizontally in outline form, or vertically as a tree structure (like an organization chart).

The development of the WBS normally occurs at the start of a project and precedes detailed project and task planning.

For this project we can refer to our Site Map in order to create an organised WBS.

Gantt chart

A Gantt chart is a type of bar chart, devised by Henry Gantt in the 1910s, that illustrates a project schedule. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements comprise the work breakdown structure of the project. Modern Gantt charts also show the dependency (i.e., precedence network) relationships between activities.

For creating a Gantt chart we have used a free software Gantt Project and using the same the following chart was created.

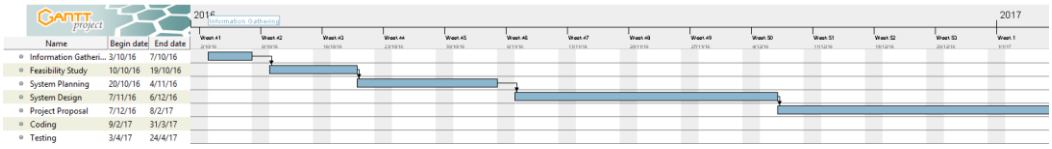


Figure 5 Project Gantt Chart 1

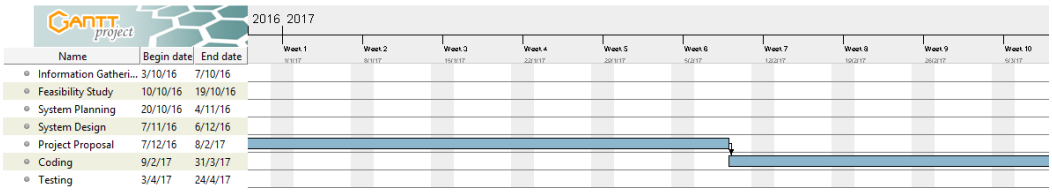


Figure 6 Project Gantt Chart 2

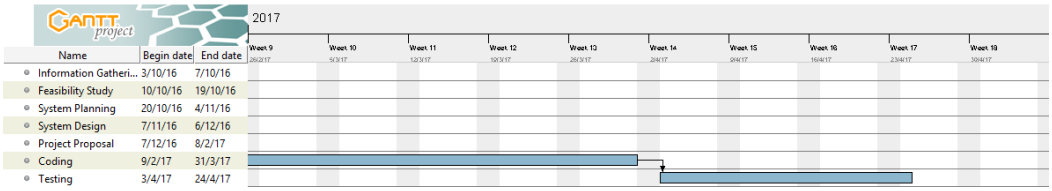


Figure 7 Project Gantt Chart 3

PERT Chart

Program Evaluation and Review Technique represents all task that are required for project's completion along with their corresponding time requirements. For a quality software, in our case a website, we require it to follow the protocols and sometimes we even create Gantt Charts to divide workload of our project into manageable time slots. A PERT chart presents a graphic illustration of a project as a network diagram consisting of numbered nodes (either circles or rectangles) representing events, or milestones in the

project linked by labelled vectors (directional lines) representing tasks in the project. The direction of the arrows on the lines indicates the sequence of tasks.

PERT is a method of analyzing the tasks involved in completing a given project, especially the time needed to complete each task, and to identify the minimum time needed to complete the total project.

PERT was developed primarily to simplify the planning and scheduling of large and complex projects. It was developed for the U.S. Navy Special Projects Office in 1957 to support the U.S. Navy's Polaris nuclear submarine project. It was able to incorporate uncertainty by making it possible to schedule a project while not knowing precisely the details and durations of all the activities. It is more of an event-oriented technique rather than start- and completion-oriented, and is used more in projects where time is the major factor rather than cost. It is applied to very large-scale, one-time, complex, non-routine infrastructure and Research and Development projects. An example of this was for the 1968 Winter Olympics in Grenoble which applied PERT from 1965 until the opening of the 1968 Games.

This project model was the first of its kind, a revival for scientific management, founded by Frederick Taylor (Taylorism) and later refined by Henry Ford (Fordism). DuPont's critical path method was invented at roughly the same time as PERT.

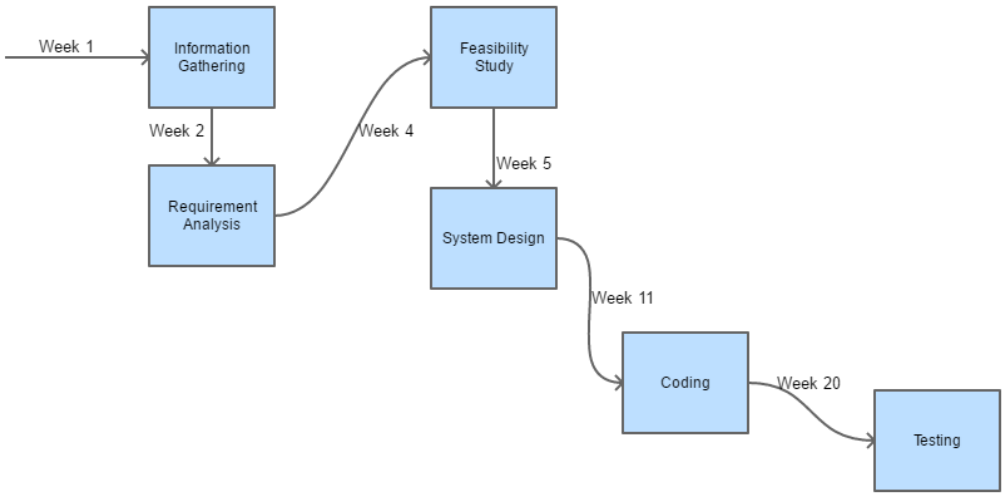


Figure 8 Pert Chart

Software requirement specification

Type of Hotels

Hotel operations vary in size, function, and cost. Most hotels and major hospitality companies that operate hotels have set widely accepted industry standards to classify hotel types. General categories include the following:

Upscale luxury:

An upscale full service hotel facility that offers luxury amenities, full service accommodations, on-site full service restaurants, and the highest level of personalized and professional service. Luxury hotels are normally classified with at least a Four Diamond or Five Diamond status or a Four or Five Star rating depending on the country and local classification standards. Examples may include: InterContinental, Waldorf Astoria, Four Seasons, Conrad, Fairmont, and The Ritz-Carlton.

Full service:

Full service hotels often contain upscale full-service facilities with a large volume of full service accommodations, on-site full service restaurants, and a variety of on-site amenities such as swimming pools, a health club, children's activities, ballrooms, on-site conference facilities, and other amenities. Examples include: Holiday Inn, Sheraton, Westin, Hilton, Marriott, and Hyatt hotels.

Historic inns and boutique hotels:

Boutique hotels are smaller independent non-branded hotels that often contain upscale facilities of varying size in unique or intimate settings with full service accommodations. Boutique hotels are generally 100 rooms or less. Some historic inns and boutique hotels may be classified as luxury hotels. Examples include Hotel Indigo and Kimpton Hotels.

Focused or select service:

Small to medium-sized hotel establishments that offer a limited number of on-site amenities that only cater and market to a specific demographic of travelers, such as the single business traveler. Most focused or select service hotels may still offer full service accommodations but may lack leisure amenities such as an on-site restaurant or a swimming pool. Examples include Crowne Plaza, Courtyard by Marriott and Hilton Garden Inn.

Economy and limited service:

Small to medium-sized hotel establishments that offer a very limited amount of on-site amenities and often only offer basic accommodations with little to no services, these facilities normally only cater and market to a specific demographic of travelers, such as the budget-minded traveler seeking a "no frills" accommodation. Limited service hotels often lack an on-site restaurant but in return may offer a limited complimentary food and beverage amenity such as on-site continental breakfast service. Examples include Ibis Budget, Hampton Inn, Aloft, Holiday Inn Express, Fairfield Inn, Four Points by Sheraton, and Days Inn.

Extended stay:

Extended stay hotels are small to medium-sized hotels that offer longer term full service accommodations compared to a traditional hotel. Extended stay hotels may offer non-traditional pricing methods such as a weekly rate that caters towards travelers in need of short-term accommodations for an extended period of time. Similar to limited and select service hotels, on-site amenities are normally limited and most extended stay hotels lack an on-site restaurant. Examples include Staybridge Suites, Candlewood Suites,

Homewood Suites by Hilton, Home2 Suites by Hilton, Residence Inn by Marriott, Element, and Extended Stay Hotels.

Timeshare and destination clubs:

Timeshare and Destination clubs are a form of property ownership also referred to as a vacation ownership involving the purchase and ownership of an individual unit of accommodation for seasonal usage during a specified period of time. Timeshare resorts often offer amenities similar that of a Full service hotel with on-site restaurant(s), swimming pools, recreation grounds, and other leisure-oriented amenities. Destination clubs on the other hand may offer more exclusive private accommodations such as private houses in a neighbourhood-style setting. Examples of timeshare brands include Hilton Grand Vacations, Marriott Vacation Club International, Westgate Resorts, Disney Vacation Club, and Holiday Inn Club Vacations.

Motel

A motel, an abbreviation for "motor hotel", is a small-sized low-rise lodging establishment similar to a limited service, lower-cost hotel, but typically with direct access to individual rooms from the car park. Motels were built to serve road travellers, including travellers on road trip vacations and workers who drive for their job (travelling salespeople, truck drivers, etc.).

New motel construction is rare in the 2000s as hotel chains have been building economy-priced, limited service franchised properties at freeway exits which compete for largely the same clientele, largely saturating the market by the 1990s. Motels are still useful in less populated areas for driving travellers, but the more populated an area becomes, the more hotels move in to meet the demand for accommodation. Many of the motels which remain in operation have joined national franchise chains, often rebranding themselves as hotels, inns or lodges.

Purpose

The Purpose of the whole process is to ease the daily or regular activities of the Hotel Management into an automatic computerized retrievable process. The daily activities includes the Room activities, Entering details of the new customer check in, To allocate a room as per the customer need and interest, Recording the checkout time and details, Releasing or Empty of room and to record the process in a computer system for future.

Due to time constraint and the minimum resources, the system is not made for the high level use. But the Management system can use the application in a very easy and minimum effort.

The aim of this Online Hotel Management System project is to build a system that will able to automate many operations in a hotel. Modern day hotels aim to create a user friendly atmosphere with the availability of concierges who remember frequent visitors and making it possible to call and make reservations. While such hotels are extremely expensive, such a service can also be provided in a cost – effective manner with the use of computers.

The process begins when a customer books a room, the booking can be placed online or through an Interactive Voice Response System (IVRS). The system will remember client preferences and can provide options accordingly. Once the room is booked, the visitor only needs to turn up and present identification. The system will also be able to send messages to the visitor’s phone to remind them of restaurants they may

have visited in the past and other options. The system can accept bookings for restaurants present in the hotel. Things – to – do around the locality can be advertised as well through messages.

Every hotel has an intricate system of cleaning and replacement of room sheets, etc. The scheduling of the same can be completely managed by this system and it can also take into account guest preferences. The guest can use the IVRS at the hotel to inform the system about the best time to clean the room.

When the guest vacates the room, the system will create the bill, schedule cleanup and allot the next guest. The customer can keep their credit card linked with the system to avoid the hassle of continually making payments. All payments in the hotel can be automatically credited from this account.

At check out the user can enter a review which will be uploaded to the website, the system will collect user reviews from other sites and from past guests, making them available to future visitors. The Online Hotel Management System will make it easy for higher level management to easily review the operation of the hotel.

- The advantages of booking a hotel online add up long before guest's arrival. The projects legendary customer service extends to the web.
- One advantage of booking with the hotel directly is the use of the hotel's full cancellation policy as well as not needing a deposit in most situations.
- Guests can read reviews and compare prices for Online Hotel Booking.
- The most important advantage of online hotel booking is convenience; guest can book room by simply sitting in home.
- Internet helps people to browse through the hotels around the world and compare the facilities and rates easily.

Scope

1. **HOTEL'S ROOM INFORMATION**: It provides User to easily search room's availability, category & easy updating of the room's records. The room numbers and cost per stay can be changed. Room's category such as deluxe, semi-deluxe can be edited and accordingly floor can be set.
2. **REPORT GENERATION**: This feature help's in easy maintenance of record of customers check-in, check-out & booking details. The reports can be generated day wise or specified timespan wise.
3. **PASSWORD PROTECTED**: This feature provides privacy to the application. The user name and password can't be identified by anybody even if somebody checks it in the database. As we are encrypting user name & passwords and storing them in the same format.
4. **DIFFERENT LOGIN LEVELS**: This feature provides different levels of Authentication.
    a. **ADMINISTRATOR**: Administrator can add and delete rooms, he can add new user, he can decide the price of the room.
    b. **STAFF**: Hotel staff can see orders in a booking and update their status as well.
    c. **USER**: User can do everything except the rights that the Administrator has
5. Some of the service providers won't allow you to choose your hotel, they only allow you to select location and quality of the hotel. Considerable discounts on hotels may be available in off-seasons.

6. Customer can utilize the serve of online hotel booking service providers when they are planning for a trip.
7. Each and every customer is capable of looking to book their hotel rooms early and conveniently.
8. User can post, update and delete the links in the all categories.
9. Online hotel booking is the best way to book rooms in favourite hotels. This facility is provided by Online Hotel Management System Project.
10. Planning a vacation has never been easier and more reasonable than now. Easiness, affordable pricing and simple comparison shopping make online hotel bookings accepted to all.

The web-site for Event Management solves all the issues thus making it better than the paper based approach.

Applicability

The website is developed using PHP and MySQL as the major components to make is easily accessible and super easy to implement by using an Apache, MySQL and PHP stack such as WAMP or XAMP for Windows machines.

 # For Apple Mac we have XAMP and for Linux machines, we can use either LAMP or XAMP.

Interfaces

Software Interfaces

The project is web based that is why we require a web server that responds to all the requests made by client. For making this project feasible and easy to develop by following the industry standards we can use the following technologies.

Apache

The Apache Software Foundation (ASF) is an American non-profit corporation to support Apache software projects, including the Apache HTTP Server. The ASF was formed from the Apache Group and incorporated in Delaware, U.S., in June 1999.

Figure 9 Apache Server Foundation

The Apache Software Foundation is a decentralized open source community of developers. The software they produce is distributed under the terms of the Apache License and is free and open source software (FOSS). The Apache projects are characterized by a collaborative, consensus-based development process and an open and pragmatic software license. Each project is managed by a self-selected team of technical experts who are active contributors to the project. The ASF is a meritocracy, implying that membership of the foundation is granted only to volunteers who have actively contributed to Apache projects. The ASF is considered a second generation open-source organization, in that commercial support is provided without the risk of platform lock-in.

Among the ASF's objectives are: to provide legal protection to volunteers working on Apache projects; to prevent the Apache brand name from being used by other organizations without permission. Installing Apache on Linux does require a bit of programming skills (though it is not too difficult). Installing it on a Windows platform is straight forward, as you can run it through a graphical user interface.

Apache's original core is fairly basic and contains a limited number of features. Its power rather comes from added functionality introduced through many modules that are written by programmers and can be installed to extend the server's capabilities. To add a new module, all you need to do is install it and restart the Apache server. Functionality that you don't need or want can easily be removed which is actually considered a good practice as it keeps the server small and light, starts faster, consumes less system resources and memory, and makes the server less prone to security holes. The Apache server also supports third party modules, some of which have been added to Apache 2 as permanent features. The Apache server very easily integrates with other open source applications, such as PHP and MySQL, making it even more powerful than it already is.

A web server in its simplest form is a computer with special software, and an internet connection that allows it to connect to other devices.

Every device connected to a network has an IP address through which others connect to and communicate with it. This IP address is sort of like a regular address that you need in real life to call or visit any contact of yours. If they didn't have an address, you wouldn't know how to call or reach them. IP addresses serve the exact same purpose. If a device didn't have one, the other machines on the same network wouldn't know how to reach it.

The Apache server offers a number of services that clients might make use of. These services are offered using various protocols through different ports, and include: hypertext transfer protocol (HTTP), typically through port 80, simple mail transfer protocol (SMTP), typically through port 25, domain name service (DNS) for mapping domain names to their corresponding IP addresses, generally through port 53, and file transfer protocol (FTP) for uploading and downloading files, usually through port 21.

How Apache Works

Apache's main role is all about communication over networks, and it uses the TCP/IP protocol (Transmission Control Protocol/Internet Protocol which allows devices with IP addresses within the same network to communicate with one another).

The TCP/IP protocol is a set of rules that define how clients make requests and how servers respond, and determine how data is transmitted, delivered, received, and acknowledged.

The Apache server is set up to run through configuration files, in which directives are added to control its behavior. In its idle state, Apache listens to the IP addresses identified in its config file (HTTPd.conf). Whenever it receives a request, it analyzes the headers, applies the rules specified for it in the Config file, and takes action.

But one server can host many websites, not just one - though, to the outside world, they seem separate from one another. To achieve this, every one of those websites has to be assigned a different name, even if those all map eventually to the same machine. This is accomplished by using what is known as virtual hosts.

Since IP addresses are difficult to remember, we, as visitors to specific sites, usually type in their respective domain names into the URL address box on our browsers. The browser then connects to a DNS server, which translates the domain names to their IP addresses. The browser then takes the returned IP address and connects to it. The browser also sends a Host header with the request so that, if the server is hosting multiple sites, it will know which one to serve back.

For example, typing in www.google.com into your browser's address field might send the following request to the server at that IP address:

```
1  GET / HTTP/1.1
2  Host: www.google.com
```

The first line contains several pieces of information. First, there is the method (in this case it's a GET), the URI, which specifies which page to be retrieved or which program to be run (in this case it's the root directory denoted by the /), and finally there is the HTTP version (which in this case is HTTP 1.1).

HTTP is a request / response stateless protocol.

HTTP is a request / response stateless protocol. It's a set of rules that govern communication between a client and the server. The client (usually but not necessarily a web browser) makes a request, the server sends back a response, and communication stops. The server doesn't look forward for more communication as is the case with other protocols that stay at a waiting state after the request is over.

If the request is successful, the server returns a 200 status code (which means that the page is found), response headers, along with the requested data. The response header of an Apache server might look something like the following:

```
01  HTTP/1.1 200 OK
02  Date: Sun, 10 Jun 2012 19:19:21 GMT
03  Server: Apache
04  Expires: Wed, 11 Jan 1984 05:00:00 GMT
05  Cache-Control: no-cache, must-revalidate, max-age=0
06  Pragma: no-cache
07  Last-Modified: Sun, 10 Jun 2012 19:19:21 GMT
08  Vary: Accept-Encoding,User-Agent
09  Content-Type: text/html; charset=UTF-8
10  Content-Length: 7560
```

The first line in the response header is the status line. It contains the HTTP version and the status code. The date follows next, and then some information about the host server and the retrieved data. The Content-Type header lets the client know the type of data retrieved so it knows how to handle it. Content-Length lets the client know the size of

the response body. If the request didn't go throw, the client would get an error code and message, such as the following response header in case of a page not found error:

```
1    HTTP/1.1 404 Not Found
```

PHP

PHP (recursive acronym for PHP: Hypertext Pre-processor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server, generating HTML which is then sent to the client. The client would receive the results of running that script, but would not know what the underlying code was. You can even configure your web server to process all your HTML files with PHP, and then there's really no way that users can tell what you have up your sleeve.

Figure 10 PHP

The best things in using PHP are that it is extremely simple for a newcomer, but offers many advanced features for a professional programmer. Don't be afraid reading the long list of PHP's features. You can jump in, in a short time, and start writing simple scripts in a few hours.

PHP code may be embedded into HTML or HTML5 markup, or it can be used in combination with various web template systems, web content management systems and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server software combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications.

The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge.

The PHP language evolved without a written formal specification or standard until 2014, leaving the canonical PHP interpreter as a de facto standard. Since 2014 work has gone on to create a formal PHP specification.

The PHP interpreter only executes PHP code within its delimiters. Anything outside its delimiters is not processed by PHP, although non-PHP text is still subject to control structures described in PHP code. The most common delimiters are <?php to open and ?> to close PHP sections. The shortened form <? also exists. This short delimiter makes script files less portable, since support for them can be disabled in the local PHP configuration and it is therefore discouraged. However, there is no recommendation against the use of the echo short tag <?=. Prior to PHP 5.4.0, this short syntax

for echo() only works with the short_open_tag configuration setting enabled, while for PHP 5.4.0 and later it is always available. The purpose of all these delimiters is to separate PHP code from non-PHP content, such as JavaScript code or HTML markup.

The first form of delimiters, <?php and ?>, in XHTML and other XML documents, creates correctly formed XML processing instructions. This means that the resulting mixture of PHP code and other mark-up in the server-side file is itself well-formed XML.

Variables are prefixed with a dollar symbol, and a type does not need to be specified in advance. PHP 5 introduced type hinting that allows functions to force their parameters to be objects of a specific class, arrays, interfaces or call back functions. However, before PHP 7.0, type hints could not be used with scalar types such as integer or string.

Unlike function and class names, variable names are case sensitive. Both double-quoted ("") and here doc strings provide the ability to interpolate a variable's value into the string. PHP treats newlines as whitespace in the manner of a free-form language, and statements are terminated by a semicolon. PHP has three types of comment syntax: /* */ marks block and inline comments; // as well as # are used for one-line comments. The echo statement is one of several facilities PHP provides to output text, e.g., to a web browser.

In terms of keywords and language syntax, PHP is similar to the C style syntax. if conditions, for and while loops, and function returns are similar in syntax to languages such as C, C++, C#, Java and Perl.

Data types

PHP stores integers in a platform-dependent range, either a 64-bit or 32-bit signed integer equivalent to the C-language long type. Unsigned integers are converted to signed values in certain situations; this behaviour is different from that of other programming languages.] Integer variables can be assigned using decimal (positive and negative), octal, hexadecimal, and binary notations.

Floating point numbers are also stored in a platform-specific range. They can be specified using floating point notation, or two forms of scientific notation.] PHP has a native Boolean type that is similar to the native Boolean types in Java and C++. Using the Boolean type conversion rules, non-zero values are interpreted as true and zero as false, as in Perl and C++.

The null data type represents a variable that has no value; NULL is the only allowed value for this data type.

Variables of the "resource" type represent references to resources from external sources. These are typically created by functions from a particular extension, and can only be processed by functions from the same extension; examples include file, image, and database resources.

Arrays can contain elements of any type that PHP can handle, including resources, objects, and other arrays. Order is preserved in lists of values and in hashes with both keys and values, and the two can be intermingled. PHP also supports strings, which can be used with single quotes, double quotes, nowdoc or heredoc syntax.

The Standard PHP Library (SPL) attempts to solve standard problems and implements efficient data access interfaces and classes.

Functions

PHP defines a large array of functions in the core language and many are also available in various extensions; these functions are well documented in the online PHP documentation. However, the built-in library has a wide variety of naming conventions and associated inconsistencies, as described under history above.

Custom functions may be defined by the developer, e.g.:

```
function myAge($birthYear) {                          // defines a function, this
one is named "myAge"
   $yearsOld = date('Y') - $birthYear;                 // calculates the age
   return $yearsOld . ' year' . ($yearsOld != 1 ? 's' : ''); // returns the age in a
descriptive form
}

echo 'I am currently ' . myAge(1981) . ' old.';            // outputs the text
concatenated
                                                  // with the return value of myAge()
// As the result of this syntax, myAge() is called.
```

In 2017, the output of the above sample program is 'I am currently 36 years old.'

In lieu of function pointers, functions in PHP can be referenced by a string containing their name. In this manner, normal PHP functions can be used, for example, as callbacks or within function tables.] User-defined functions may be created at any time without being prototyped. Functions may be defined inside code blocks, permitting a run-time decision as to whether or not a function should be defined. There is a function_exists function that determines whether a function with a given name has already been defined. Function calls must use parentheses, with the exception of zero-argument class constructor functions called with the PHP operator new, in which case parentheses are optional.

Until PHP 5.3, support for anonymous functions and closures did not exist in PHP. While create_function() exists since PHP 4.0.1, it is merely a thin wrapper around eval()that allows normal PHP functions to be created during program execution. PHP 5.3 added syntax to define an anonymous function or "closure" which can capture variables from the surrounding scope:

```
function getAdder($x) {
   return function($y) use ($x) {
      return $x + $y;
   };
}

$adder = getAdder(8);
echo $adder(2); // prints "10"
```

In the example above, getAdder() function creates a closure using passed argument $x (the keyword use imports a variable from the lexical context), which takes an additional argument $y, and returns the cr eated closure to the caller. Such a function is a first-class object, meaning that it can be stored in a variable, passed as a parameter to other functions, etc.

Unusually for a dynamically typed language, PHP supports type declarations on function parameters, which are enforced at runtime. This has been supported for classes and interfaces since PHP 5.0, for arrays since PHP 5.1, for "callables" since PHP 5.4, and scalar (integer, float, string and boolean) types since PHP 7.0. PHP 7.0 also has type declarations for function return types, expressed by placing the type name after the list of parameters, preceded by a colon. For example, the getAdder function from the earlier example could be annotated with types like so in PHP 7:

```
function getAdder(int $x): \Closure {
   return function(int $y) use ($x) : int {
      return $x + $y;
   };
}


$adder = getAdder(8);
echo $adder(2);       // prints "10"
echo $adder(null);    // throws an exception because an incorrect type was
passed
$adder = getAdder([]); // would also throw an exception
```

By default, scalar type declarations follow weak typing principles. So, for example, if a parameter's type is int, PHP would allow not only integers, but also convertible numeric strings, floats or Booleans to be passed to that function, and would convert them. However, PHP 7 has a "strict typing" mode which, when used, disallows such conversions for function calls and returns within a file.

Object-oriented programming

Basic object-oriented programming functionality was added in PHP 3 and improved in PHP 4. This allowed for PHP to gain further abstraction, making creative tasks easier for programmers using the language. Object handling was completely rewritten for PHP 5, expanding the feature set and enhancing performance. In previous versions of PHP, objects were handled like value types. The drawback of this method was that code had to make heavy use of PHP's "reference" variables if it wanted to modify an object it was passed rather than creating a copy of it. In the new approach, objects are referenced by handle, and not by value.

PHP 5 introduced private and protected member variables and methods, along with abstract classes, final classes, abstract methods, and final methods. It also introduced a standard way of declaring constructors and destructors, similar to that of other object-oriented languages such as C++, and a standard exception handling model. Furthermore, PHP 5 added interfaces and allowed for multiple interfaces to be implemented. There are special interfaces that allow objects to interact with the runtime system. Objects implementing Array Access can be used with array syntax and objects implementing Iterator or Iterator Aggregate can be used with the for each language construct. There is no virtual table feature in the engine, so static variables are bound with a name instead of a reference at compile time.

If the developer creates a copy of an object using the reserved word clone, the Zend engine will check whether a __clone() method has been defined. If not, it will call a default __clone() which will copy the object's properties. If a __clone() method is defined, then it will be responsible for setting the necessary properties in the created object. For convenience, the engine will supply a function that imports the properties of the source object, so the programmer can start with a by-value replica of the source object and only override properties that need to be changed.

The following is a basic example of object-oriented programming in PHP:

```
class Person
{
   public $firstName;
   public $lastName;

   public function __construct($firstName, $lastName = '') { // optional second
argument
      $this->firstName = $firstName;
      $this->lastName  = $lastName;
   }

   public function greet() {
      return 'Hello, my name is ' . $this->firstName .
           (($this->lastName != '') ? (' ' . $this->lastName) : '') . '.';
   }

   public static function staticGreet($firstName, $lastName) {
      return 'Hello, my name is ' . $firstName . ' ' . $lastName . '.';
   }
}

$he    = new Person('John', 'Smith');
$she   = new Person('Sally', 'Davis');
$other = new Person('iAmine');

echo $he->greet(); // prints "Hello, my name is John Smith."
echo '<br />';

echo $she->greet(); // prints "Hello, my name is Sally Davis."
echo '<br />';

echo $other->greet(); // prints "Hello, my name is iAmine."
echo '<br />';

echo Person::staticGreet('Jane', 'Doe'); // prints "Hello, my name is Jane Doe."
```

The visibility of PHP properties and methods is defined using
the keywords public, private, and protected. The default is public, if only var is

used; var is a synonym for public. Items declared public can be accessed everywhere. protected limits access to inherited classes (and to the class that defines the item). private limits visibility only to the class that defines the item. Objects of the same type have access to each other's private and protected members even though they are not the same instance. PHP's member visibility features have sometimes been described as "highly useful." However, they have also sometimes been described as "at best irrelevant and at worst positively harmful."

Implementations:

The original, only complete and most widely used PHP implementation is powered by the Zend Engine and known simply as PHP. To disambiguate it from other implementations, it is sometimes unofficially referred to as "Zend PHP". The Zend Engine compiles PHP source code on-the-fly into an internal format that it can execute, thus it works as an interpreter. It is also the "reference implementation" of PHP, as PHP has no formal specification, and so the semantics of Zend PHP define the semantics of PHP itself. Due to the complex and nuanced semantics of PHP, defined by how Zend works, it is difficult for competing implementations to offer complete compatibility.

PHP's single-request-per-script-execution model, and the fact that the Zend Engine is an interpreter, leads to inefficiency; as a result, various products have been developed to help improve PHP performance. In order to speed up execution time and not have to compile the PHP source code every time the web page is accessed, PHP scripts can also be deployed in the PHP engine's internal format by using an opcode cache, which works by caching the compiled form of a PHP script (opcodes) in shared memory to avoid the overhead of parsing and compiling the code every time the script runs. An opcode cache, Zend Opcache, is built into PHP since version 5.5. Another example of a widely used opcode cache is the Alternative PHP Cache (APC), which is available as a PECL extension.

While Zend PHP is still the most popular implementation, several other implementations have been developed. Some of these are compilers or support JIT compilation, and hence offer performance benefits over Zend PHP at the expense of lacking full PHP compatibility. Alternative implementations include the following:

HipHop Virtual Machine (HHVM) – developed at Facebook and available as open source, it converts PHP code into a high-level bytecode (commonly known as an intermediate language), which is then translated into x86-64 machine code dynamically at runtime by a just-in-time (JIT) compiler, resulting in up to 6× performance improvements.

Parrot – a virtual machine designed to run dynamic languages efficiently; Pipp transforms the PHP source code into the Parrot intermediate representation, which is then translated into the Parrot's bytecode and executed by the virtual machine.

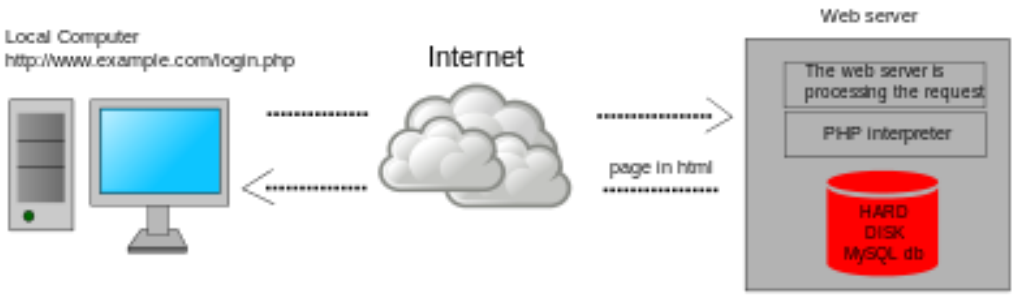
Phalanger – compiles PHP into Common Intermediate Language (CIL) bytecode

HipHop – developed at Facebook and available as open source, it transforms the PHP scripts into C++ code and then compiles the resulting code, reducing the server load up to 50%. In early 2013, Facebook deprecated it in favor of HHVM due to multiple reasons, including deployment difficulties and lack of support for the whole PHP language, including the create_function() and eval() constructs.

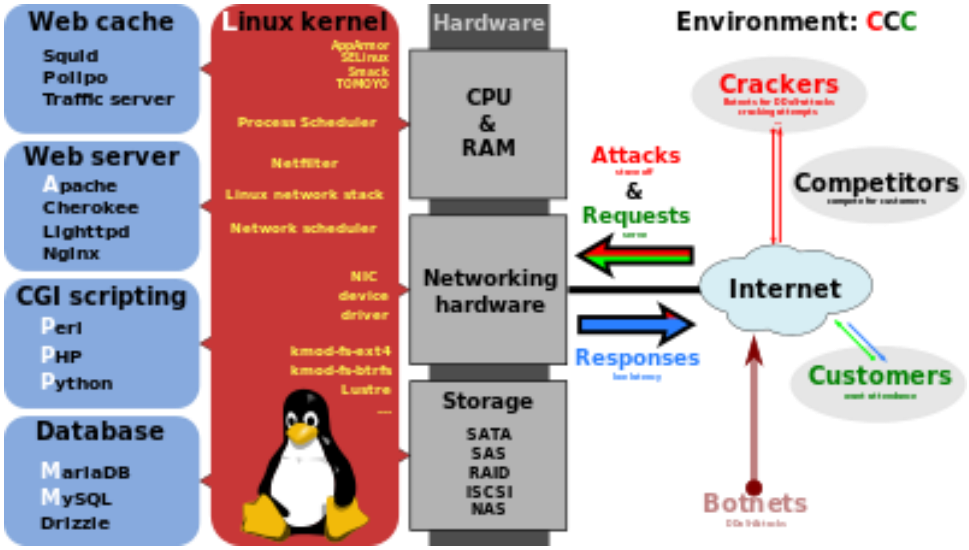
Use

A broad overview of the LAMP software bundle, displayed here together with Squid.

PHP is a general-purpose scripting language that is especially suited to server-side web development, in which case PHP generally runs on a web server. Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic web page content or dynamic images used on websites or else where.It can also be used for command-line scripting and client-side graphical user interface (GUI) applications. PHP can be deployed on most web servers, many operating systems and platforms, and can be used with many relational database management systems (RDBMS). Most web hosting providers support PHP for use by their clients. It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use.



Dynamic web page: example of server-side scripting (PHP and MySQL).



PHP acts primarily as a filter, taking input from a file or stream containing text and/or PHP instructions and outputting another stream of data. Most commonly the output will be HTML, although it could be JSON, XML or binary data such as image or audio formats. Since PHP 4, the PHP parser compiles input to produce bytecode for processing by the Zend Engine, giving improved performance over its interpreter predecessor.

Originally designed to create dynamic web pages, PHP now focuses mainly on server-side scripting, and it is similar to other server-side scripting languages that provide dynamic content from a web server to a client, such as Microsoft's ASP.NET, Sun Microsystems' Java Server Pages, and mod_perl. PHP has also attracted the

development of many software frameworks that provide building blocks and a design structure to promote rapid application development (RAD). Some of these include PRADO, CakePHP, Symfony, CodeIgniter, Laravel, Yii Framework, Phalcon and Zend Framework, offering features similar to other web frameworks.

The LAMP architecture has become popular in the web industry as a way of deploying web applications. PHP is commonly used as the P in this bundle alongside Linux, Apache and MySQL, although the P may also refer to Python, Perl, or some mix of the three. Similar packages, WAMP and MAMP, are also available for Windows and macOS, with the first letter standing for the respective operating system. Although both PHP and Apache are provided as part of the macOS base install, users of these packages seek a simpler installation mechanism that can be more easily kept up to date.

As of April 2007, over 20 million Internet domains had web services hosted on servers with PHP installed and mod_php was recorded as the most popular Apache HTTP Server module. As of October 2010, PHP was used as the server-side programming language on 75% of all websites whose server-side programming language was known (as of February 2014, the percentage had reached 82%), and PHP was the most-used open source software within enterprises. Web content management systems written in PHP include MediaWiki, Joomla, eZ Publish, eZ Platform, SilverStripe, WordPress, Drupal, and Moodle. Websites written in PHP, in back-end and/or user-facing portion, include Facebook, Digg, Tumblr, Dailymotion, and Slack.

For specific and more advanced usage scenarios, PHP offers a well defined and documented way for writing custom extensions in C or C++. Besides extending the language itself in form of additional libraries, extensions are providing a way for improving execution speed where it is critical and there is room for improvements by using a true compiled language. PHP also offers well defined ways for embedding itself into other software projects. That way PHP can be easily used as an internal scripting language for another project, also providing tight interfacing with the project's specific internal data structures.

PHP received mixed reviews due to lacking support for multithreading at the core language level, though using threads is made possible by the "pthreads" PECL extension.

As of January 2013, PHP was used in more than 240 million websites (39% of those sampled) and was installed on 2.1 million web servers.

HTML5

HTML5 is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and current version of the HTML standard.

It was published in October 2014 by the World Wide Web Consortium (W3C) to improve the language with support for the latest multimedia, while keeping it both easily readable by humans and consistently understood by computers and devices such as web browsers, parsers, etc. HTML5 is intended to subsume not only HTML 4, but also XHTML 1 and DOM Level 2 HTML.

Figure 11 HTML 5

HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications. For the same reasons, HTML5 is also a candidate for cross-platform mobile applications, because it includes features designed with low-powered devices in mind.

Many new syntactic features are included. To natively include and handle multimedia and graphical content, the new <video>, <audio> and <canvas> elements were added, and support for scalable vector graphics (SVG) content and MathML for mathematical formulas. To enrich the semantic content of documents, new page structure elements such as <main>, <section>, <article>, <header>, <footer>, <aside>, <nav> and <figure>, are added. New attributes are introduced, some elements and attributes have been removed, and others such as <a>, <cite> and <menu> have been changed, redefined or standardized.

The APIs and Document Object Model (DOM) are now fundamental parts of the HTML5 specification and HTML5 also better defines the processing for any invalid documents.

Features And APIs

The W3C proposed a greater reliance on modularity as a key part of the plan to make faster progress, meaning identifying specific features, either proposed or already existing in the spec, and advancing them as separate specifications. Some technologies that were originally defined in HTML5 itself are now defined in separate specifications:

HTML Working Group – HTML Canvas 2D Context;

Web Apps Working Group – Web Messaging, Web Workers, Web Storage, WebSocket, Server-sent events, Web Components (this was not part of HTML5 though); Note that the Web Applications Working Group was closed in October 2015 and its deliverables transferred to the Web Platform Working Group (WPWG).

IETF HyBi Working Group – Web Socket Protocol;

Web RTC Working Group – WebRTC;

Web Media Text Tracks Community Group – WebVTT.

After the standardization of the HTML5 specification in October 2014, the core vocabulary and features are being extended in four ways. Likewise, some features that were removed from the original HTML5 specification have been standardized separately as modules, such as Microdata and Canvas. Technical specifications introduced as HTML5 extensions such as Polyglot Markup have also been standardized as modules. Some W3C specifications that were originally separate specifications have been adapted as HTML5 extensions or features, such as SVG. Some features that might have slowed

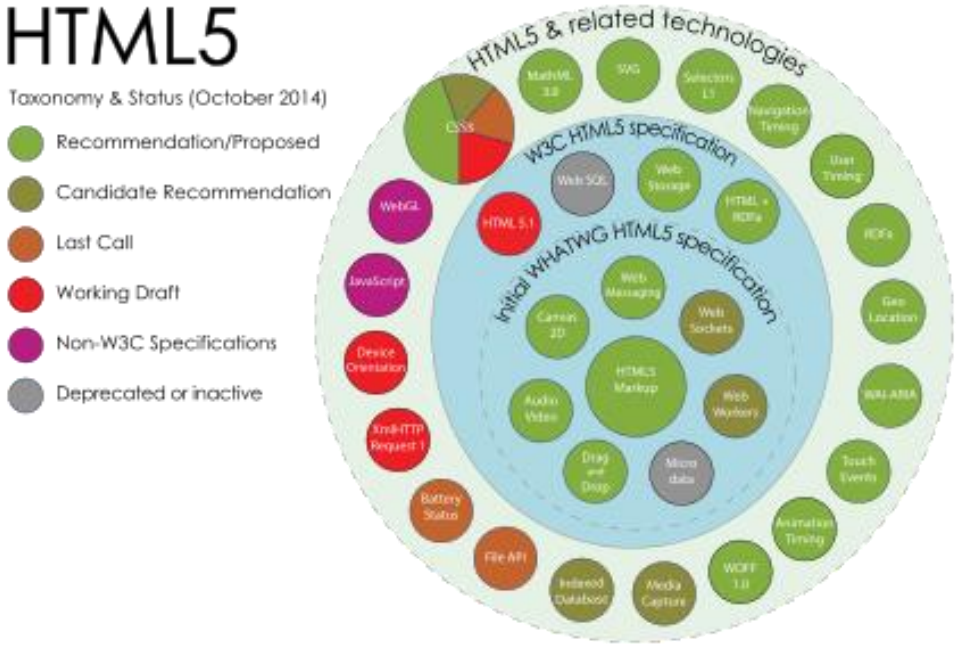
down the standardization of HTML5 will be standardized as upcoming specifications, instead. HTML 5.1 is expected to be finalized in 2016, and it is currently on the standardization track at the W3C.

FEATURES

Mark-up: HTML5 introduces elements and attributes that reflect typical usage on modern websites. Some of them are semantic replacements for common uses of generic block ( ) and inline () elements, for example <nav> (website navigation block), <footer> (usually referring to bottom of web page or to last lines of HTML code), or <audio>and <video> instead of <object>. Some deprecated elements from HTML 4.01 have been dropped, including purely presentational elements such as <font> and , whose effects have long been superseded by the more capable Cascading Style Sheets. There is also a renewed emphasis on the importance of DOM scripting (e.g., JavaScript) in Web behavior.

The HTML5 syntax is no longer based on SGML despite the similarity of its markup. It has, however, been designed to be backward compatible with common parsing of older versions of HTML. It comes with a new introductory line that looks like an SGML document type declaration, <!DOCTYPE html>, which triggers the standards-compliant rendering mode.https://en.wikipedia.org/wiki/HTML5 Since 5 January 2009, HTML5 also includes Web Forms 2.0, a previously separate WHATWG specification. Since 5 January 2009, HTML5 also includes Web Forms 2.0, a previously separate WHATWG specification.

New APIs



HTML5 related APIs

In addition to specifying markup, HTML5 specifies scripting application programming interfaces (APIs) that can be used with JavaScript. Existing document object model (DOM) interfaces are extended and de facto features documented. There are also new APIs, such as:

- Canvas;
- Timed Media Playback;
- Offline;
- Editable content;
- Drag-and-drop
- History
- MIME type and protocol handler registration;
- Microdata;
- Web Messaging;

Web Storage – a key-value pair storage framework that provides behaviour similar to cookies but with larger storage capacity and improved API.

Not all of the above technologies are included in the W3C HTML5 specification, though they are in the WHATWG HTML specification. Some related technologies, which are not part of either the W3C HTML5 or the WHATWG HTML specification, are as follows. The W3C publishes specifications for these separately:

Geolocation

Web SQL Database – a local SQL Database (no longer maintained);

Indexed DB – an indexed hierarchical key-value store (formerly Web Simple DB);

File – an API intended to handle file uploads and file manipulation;

Directories and System – an API intended to satisfy client-side-storage use cases not well served by databases;

File Writer – an API for writing to files from web applications;

Web Audio – a high-level JavaScript API for processing and synthesizing audio in web applications;

Class List.

Web Cryptography

Web RTC

HTML5 cannot provide animation within web pages. Additional JavaScript or CSS3 functionality is necessary for animating HTML elements. Animation is also possible using JavaScript and HTML 4, and within SVG elements through SMIL, although browser support of the latter remains uneven as of 2011.

XHTML5 (XML-serialized HTML5)

XML documents must be served with an XML Internet media type (often called "MIME type") such as application/xhtml+xml or application/xml, and must conform to strict, well-formed syntax of XML. XHTML5 is simply XML-serialized HTML5 data (e.g. not having any unclosed tags), sent with one of XML media types. HTML that has been written to conform to both the HTML and XHTML specifications – and which will therefore produce the same DOM tree whether parsed as HTML or XML – is called polyglot markup.

Error handling

HTML5 is designed so that old browsers can safely ignore new HTML5 constructs. In contrast to HTML 4.01, the HTML5 specification gives detailed rules for lexing and parsing, with the intent that compliant browsers will produce the same results when parsing incorrect syntax.https://en.wikipedia.org/wiki/HTML5 Although HTML5 now defines a consistent behavior for "tag soup" documents, those documents are not regarded as conforming to the HTML5 standard. Although HTML5 now defines a consistent behavior for "tag soup" documents, those documents are not regarded as conforming to the HTML5 standard.

Popularity

According to a report released on 30 September 2011, 34 of the world's top 100 Web sites were using HTML5 – the adoption led by search engines and social networks. Another report released in August 2013 has shown that 153 of the Fortune 500 U.S. companies implemented HTML5 on their corporate websites.

Since 2014, HTML5 is at least partially supported by most popular layout engines.

Differences from HTML 4.01 and XHTML 1.x

The following is a cursory list of differences and some specific examples.

New parsing rules: oriented towards flexible parsing and compatibility; not based on SGML

Ability to use inline SVG and MathML in text/html

New elements: article, aside, audio, bdi, canvas, command, data, datalist, details, embed, figcaption, figure, footer, header, keygen, mark, meter, nav, output, progress, rp, rt, rub y, section, source, summary, time, track, video, wbr

New types of form controls: dates and times, email, url, search, number, range, tel, color

New attributes: charset (on meta), a sync (on script)

Global attributes (that can be applied for every element): id, tab index, hidden, data-* (custom data attributes)

Deprecated elements will be dropped altogether: acronym, applet, basefont, big, center, dir, font, frame, frameset, isindex, nof rames, strike, tt

dev.w3.org provides the latest Editors Draft of "HTML5 differences from HTML 4", which provides a complete outline of additions, removals and changes between HTML5 and HTML 4.

CSS3

CSS3 offers a huge variety of new ways to create an impact with your designs, with quite a few important changes. This first tutorial will give you a very basic introduction to the new possibilities created by the standard.

Figure 12 CSS3

Modules in CSS3

The development of CSS3 is going to be split up into ‘modules’. The old specification was simply too large and complex to be updated as one, so it has been broken down into smaller pieces – with new ones also added. Some of these modules include:

- The Box Model
- Lists Module
- Hyperlink Presentation
- Speech Module
- Backgrounds and Borders
- Text Effects
- Multi-Column Layout

Several of the modules have now been completed, including SVG (Scalable Vector Graphics), Media Queries and Namespaces. The others are still being worked upon.

It is incredibly difficult to give a projected date when web browsers will adopt the new features of CSS3 – some new builds of Safari have already started to.

New features will be implemented gradually in different browsers, and it could still be a year or two before every module is widely adopted.

Hopefully, in a mainly positive way. CSS3 will obviously be completely backwards compatible, so it won't be necessary to change existing designs to ensure they work – web browsers will always continue to support CSS2.

The main impact will be the ability to use new selectors and properties which are available. These will allow you to both achieve new design features (animation or gradients for instance), and achieve current design features in a much easier way (e.g. using columns).

CSS is designed primarily to enable the separation of document content from document presentation, including aspects such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Separation of formatting and content makes it possible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. It can also display the web page differently depending on the screen size or viewing device. Readers can also specify a different style sheet, such as a CSS file stored on their own computer, to override the one the author specified.

Changes to the graphic design of a document (or hundreds of documents) can be applied quickly and easily, by editing a few lines in the CSS file they use, rather than by changing markup in the documents.

The CSS specification describes a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities (or weights) are calculated and assigned to rules, so that the results are predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.

Syntax

CSS has a simple syntax and uses a number of English keywords to specify the names of various style properties.

A style sheet consists of a list of rules. Each rule or rule-set consists of one or more selectors, and a declaration block.

Selector

In CSS, selectors declare which part of the markup a style applies to by matching tags and attributes in the markup itself.

Selectors may apply to:

- all elements of a specific type, e.g. the second-level headers h2
- elements specified by attribute, in particular:
- id: an identifier unique within the document

- class: an identifier that can annotate multiple elements in a document
- Elements depending on how they are placed relative to others in the document tree.

Classes and IDs are case-sensitive, start with letters, and can include alphanumeric characters and underscores. A class may apply to any number of instances of any elements. An ID may only be applied to a single element.

Pseudo-classes are used in CSS selectors to permit formatting based on information that is not contained in the document tree. One example of a widely used pseudo-class is :hover, which identifies content only when the user "points to" the visible element, usually by holding the mouse cursor over it. It is appended to a selector as in a:hover or #elementid:hover. A pseudo-class classifies document elements, such as :link or :visited, whereas a pseudo-element makes a selection that may consist of partial elements, such as ::first-line or ::first-letter.

Selectors may be combined in many ways to achieve great specificity and flexibility.[6] Multiple selectors may be joined in a spaced list to specify elements by location, element type, id, class, or any combination thereof. The order of the selectors is important. For example, div .myClass {color: red;} applies to all elements of class myClass that are inside div elements, whereas .myClass div {color: red;} applies to all div elements that are in elements of class myClass.

Use

Before CSS, nearly all presentational attributes of HTML documents were contained within the HTML markup. All font colors, background styles, element alignments, borders and sizes had to be explicitly described, often repeatedly, within the HTML. CSS lets authors move much of that information to another file, the style sheet, resulting in considerably simpler HTML.

For example, headings (h1 elements), sub-headings (h2), sub-sub-headings (h3), etc., are defined structurally using HTML. In print and on the screen, choice of font, size, color and emphasis for these elements is presentational.

Before CSS, document authors who wanted to assign such typographic characteristics to, say, all h2 headings had to repeat HTML presentational markup for each occurrence of that heading type. This made documents more complex, larger, and more error-prone and difficult to maintain. CSS allows the separation of presentation from structure. CSS can define color, font, text alignment, size, borders, spacing, layout and many other typographic characteristics, and can do so independently for on-screen and printed views. CSS also defines non-visual styles, such as reading speed and emphasis for aural text readers. The W3C has now deprecated the use of all presentational HTML markup.

For example, under pre-CSS HTML, a heading element defined with red text would be written as:

```
<h1><font color="red"> Chapter 1. </font></h1>
```

Using CSS, the same element can be coded using style properties instead of HTML presentational attributes:

```
<h1 style="color: red;"> Chapter 1. </h1>
```

An "external" CSS file, as described below, can be associated with an HTML document using the following syntax:

```
<link href="path/to/file.css" rel="stylesheet" type="text/css">
```

An internal CSS code can be typed in the head section of the code. The coding is started with the style tag. For example,

```
<style>
h1 {color: red;}
</style>
```

Limitations

Some noted limitations of the current capabilities of CSS include:

Selectors are unable to ascend

CSS currently offers no way to select a parent or ancestor of an element that satisfies certain criteria. CSS Selectors Level 4, which is still in Working Draft status, proposes such a selector, but only as part of the "complete" selector profile, not the "fast" profile used in dynamic CSS styling. A more advanced selector scheme (such as XPath) would enable more sophisticated style sheets. The major reasons for the CSS Working Group previously rejecting proposals for parent selectors are related to browser performance and incremental rendering issues.

Cannot explicitly declare new scope independently of position

Scoping rules for properties such as z-index look for the closest parent element with a position: absolute or position: relative attribute. This odd coupling has undesired effects. For example, it is impossible to avoid declaring a new scope when one is forced to adjust an element's position, preventing one from using the desired scope of a parent element.

Pseudo-class dynamic behaviour not controllable

CSS implements pseudo-classes that allow a degree of user feedback by conditional application of alternate styles. One CSS pseudo-class, ":hover", is dynamic (equivalent of JavaScript "onmouseover") and has potential for abuse (e.g., implementing cursor-proximity popups),https://en.wikipedia.org/wiki/Cascading_Style_Sheets but CSS has no ability for a client to disable it (no "disable"-like property) or limit its effects (no "no change"-like values for each property).

Cannot name rules

There is no way to name a CSS rule, which would allow (for example) client-side scripts to refer to the rule even if its selector changes.

Cannot include styles from a rule into another rule

CSS styles often must be duplicated in several rules to achieve a desired effect, causing additional maintenance and requiring more thorough testing. Some new CSS features were proposed to solve this, but (as of February, 2016) are not yet implemented anywhere.

Cannot target specific text without altering markup

Besides the :first-letter pseudo-element, one cannot target specific ranges of text without needing to utilize place-holder elements.

Resolved Limitations

Vertical control limitations

Though horizontal placement of elements was always generally easy to control, vertical placement was frequently unintuitive, convoluted, or outright impossible. Simple tasks, such as centering an element vertically or placing a footer no higher than bottom of the viewport required either complicated and unintuitive style rules, or simple but widely unsupported rules. The Flexible Box Module improved the situation considerably and vertical control is much more straightforward and supported in all of the modern browsers. Older browsers still have those issues, but most of those (mainly Internet Explorer 9 and below) are no longer supported by their vendors.

Absence of expressions

There was no standard ability to specify property values as simple expressions (such as margin-left: 10% – 3em + 4px;). This would be useful in a variety of cases, such as calculating the size of columns subject to a constraint on the sum of all columns. Internet Explorer versions 5 to 7 support a proprietary expression() statement,with similar functionality. This proprietary expression() statement is no longer supported from Internet Explorer 8 onwards, except in compatibility modes. This decision was taken for "standards compliance, browser performance, and security reasons". However, a candidate recommendation with a calc() value to address this limitation has been published by the CSS WG and has since been supported in all of the modern browsers.

Lack of column declaration

Although possible in current CSS 3 (using the column-count module), layouts with multiple columns can be complex to implement in CSS 2.1. With CSS 2.1, the process is often done using floating elements, which are often rendered differently by different browsers, different computer screen shapes, and different screen ratios set on standard monitors. All of the modern browsers support this CSS 3 feature in one form or another.

Advantages

Separation of content from presentation

CSS facilitates publication of content in multiple presentation formats based on nominal parameters. Nominal parameters include explicit user preferences, different web browsers, the type of device being used to view the content (a desktop computer or mobile Internet device), the geographic location of the user and many other variables.

Site-wide consistency

When CSS is used effectively, in terms of inheritance and "cascading", a global style sheet can be used to affect and style elements site-wide. If the situation arises that the styling of the elements should be changed or adjusted, these changes can be made by editing rules in the global style sheet. Before CSS, this sort of maintenance was more difficult, expensive and time-consuming.

Bandwidth

A stylesheet, internal or external, specifies the style once for a range of HTML elements selected by class, type or relationship to others. This is much more efficient than repeating style information inline for each occurrence of the element. An external stylesheet is usually stored in the browser cache, and can therefore be used on multiple pages without being reloaded, further reducing data transfer over a network.

Page reformatting

With a simple change of one line, a different style sheet can be used for the same page. This has advantages for accessibility, as well as providing the ability to tailor a page or site to different target devices. Furthermore, devices not able to understand the styling still display the content.

Accessibility

Without CSS, web designers must typically lay out their pages with techniques such as HTML tables that hinder accessibility for vision-impaired users (see Tableless web design#Accessibility).

CSS Framework

CSS frameworks are pre-prepared libraries that are meant to allow for easier, more standards-compliant styling of web pages using the Cascading Style Sheets language. CSS frameworks include Foundation, Blueprint, Bootstrap, Cascade Framework and Materialize. Like programming and scripting language libraries, CSS frameworks are usually incorporated as external .css sheets referenced in the HTML <head>. They provide a number of ready-made options for designing and laying out the web page. Although many of these frameworks have been published, some authors use them mostly for rapid prototyping, or for learning from, and prefer to 'handcraft' CSS that is appropriate to each published site without the design, maintenance and download overhead of having many unused features in the site's styling.

Twitter Bootstrap3

Bootstrap is a powerful front-end framework for faster and easier web development. It includes HTML and CSS based design templates for common user interface components like Typography, Forms, Buttons, Tables, Navigations, Dropdowns, Alerts, Modals, Tabs, Accordion, Carousel and many other as well as optional JavaScript extensions.

Bootstrap, originally named Twitter Blueprint, was developed by Mark Otto and Jacob Thornton at Twitter as a framework to encourage consistency across internal tools. Before Bootstrap, various libraries were used for interface development, which led to inconsistencies and a high maintenance burden. According to Twitter developer Mark Otto:

"A super small group of developers and I got together to design and build a new internal tool and saw an opportunity to do something more. Through that process, we saw ourselves build something much more substantial than another internal tool. Months later, we ended up with an early version of Bootstrap as a way to document and share common design patterns and assets within the company."

Figure 13 Bootstrap 3

After a few months of development by a small group, many developers at Twitter began to contribute to the project as a part of Hack Week, a hackathon-style week for the Twitter development team. It was renamed from Twitter Blueprint to Bootstrap, and

released as an open source project on August 19, 2011. It has continued to be maintained by Mark Otto, Jacob Thornton, and a small group of core developers, as well as a large community of contributors.

On January 31, 2012, Bootstrap 2 was released, which added a twelve-column responsive grid layout system, inbuilt support for Glyph icons, several new components, as well as changes to many of the existing components.

On August 19, 2013, Bootstrap 3 was released, which redesigned components to use flat design, and a mobile first approach.

On October 29, 2014, Mark Otto announced that Bootstrap 4 was in development. The first alpha version of Bootstrap 4 was released on August 19, 2015.

Structure and Functions

Bootstrap is modular and consists of a series of Less stylesheets that implement the various components of the toolkit. These stylesheets are generally compiled into a bundle and included in web pages, but individual components can be included or removed. Bootstrap provides a number of configuration variables that control things such as color and padding of various components.

Since Bootstrap 2, the Bootstrap documentation has included a customization wizard which generates a customized version of Bootstrap based on the requested components and various settings.

As of Bootstrap 4, Sass is used instead of Less for the stylesheets.

Each Bootstrap component consists of an HTML structure, CSS declarations, and in some cases accompanying JavaScript code.

Grid system and responsive design comes standard with an 1170 pixel wide grid layout. Alternatively, the developer can use a variable-width layout. For both cases, the toolkit has four variations to make use of different resolutions and types of devices: mobile phones, portrait and landscape, tablets and PCs with low and high resolution. Each variation adjusts the width of the columns.

Stylesheets

Bootstrap provides a set of stylesheets that provide basic style definitions for all key HTML components. These provide a uniform, modern appearance for formatting text, tables and form elements.

Re-usable components

In addition to the regular HTML elements, Bootstrap contains other commonly used interface elements. The components are implemented as CSS classes, which must be applied to certain HTML elements in a page.

JavaScript components

Bootstrap comes with several JavaScript components in the form of jQuery plugins. They provide additional user interface elements such as dialog boxes, tooltips, and carousels. They also extend the functionality of some existing interface elements, including for example an auto-complete function for input fields. In version 2.0, the

following JavaScript plugins are supported: Modal, Dropdown, Scroll spy, Tab, Tooltip, Popover, Alert, Button, Collapse, Carousel and Type ahead.

Bootstrap also gives you ability to create responsive layout with much less efforts.

Advantages of Bootstrap

The biggest advantage of using Bootstrap is that it comes with free set of tools for creating flexible and responsive web layouts as well as common interface components.

Additionally, using the Bootstrap data APIs you can create advanced interface components like Scroll spy and Type heads without writing a single line of JavaScript.

Here are some more advantages, why one should opt for Bootstrap:

Save lots of time

You can save lots of time and efforts using the Bootstrap predefined design templates and classes and concentrate on other development work.

Responsive features

Using Bootstrap you can easily create responsive designs. Bootstrap responsive features make your web pages to appear more appropriately on different devices and screen resolutions without any change in mark-up.

Consistent design

All Bootstrap components share the same design templates and styles through a central library, so that the designs and layouts of your web pages are consistent throughout your development.

Easy to use

Bootstrap is very easy to use. Anybody with the basic working knowledge of HTML and CSS can start development with Bootstrap.

Compatible with browsers

Bootstrap is created with modern browsers in mind and it is compatible with all modern browsers such as Mozilla Firefox, Google Chrome, Safari, Internet Explorer, and Opera.

Open Source

The best part is, it is completely free to download and use.

LESS

Less (sometimes stylized as LESS) is a dynamic style sheet language that can be compiled into Cascading Style Sheets (CSS) and run on the client side or server side. Designed by Alexis Sellier, Less is influenced by Sass and has influenced the newer "SCSS" syntax of Sass, which adapted its CSS-like block formatting syntax. Less is open source.

Figure 14 LESS (Less CSS)

Its first version was written in Ruby; however, in the later versions, use of Ruby has been deprecated and replaced by JavaScript. The indented syntax of Less is a nested metalanguage, as valid CSS is valid Less code with the same semantics. Less provides the following mechanisms: variables, nesting, mixes, operators and functions; the main

difference between Less and other CSS precompiles being that Less allows real-time compilation via less.js by the browser.

Variables

These are pretty self-explanatory:

```
@nice-blue: #5B83AD;
@light-blue: @nice-blue + #111;
#header {
  color: @light-blue;
}
```

Outputs

```
#header {
  color: #6c94be;
}
```

Note that variables are actually "constants" in that they can only be defined once.

Mixins

Mixins are a way of including ("mixing in") a bunch of properties from one rule-set into another rule-set. So say we have the following class:

```
.bordered {
  border-top: dotted 1px black;
  border-bottom: solid 2px black;
}
```

And we want to use these properties inside other rule-sets. Well, we just have to drop in the name of the class where we want the properties, like so:

```
#menu a {
  color: #111;
  .bordered;
}
.post a {
  color: red;
  .bordered;
}
```

The properties of the .bordered class will now appear in both #menu a and .post a. (Note that you can also use #ids as mixins.)

Nested Rules

Less gives you the ability to use nesting instead of, or in combination with cascading. Let's say we have the following CSS:

```
#header {
  color: black;
}
#header .navigation {
  font-size: 12px;
}
```

```
#header .logo {
  width: 300px;
}
```

In Less, we can also write it this way:

```
#header {
  color: black;
  .navigation {
    font-size: 12px;
  }
  .logo {
    width: 300px;
  }
}
```

The resulting code is more concise, and mimics the structure of your HTML.

You can also bundle pseudo-selectors with your mixins using this method. Here's the classic clearfix hack, rewritten as a mixin (& represents the current selector parent):

```
.clearfix {
  display: block;
  zoom: 1;
  &:after {
    content: " ";
    display: block;
    font-size: 0;
    height: 0;
    clear: both;
    visibility: hidden;
  }
}
```

Nested Directives and Bubbling

Directives such as media or keyframe can be nested in the same way as selectors. Directive is placed on top and relative order against other elements inside the same ruleset remains unchanged. This is called bubbling.

Conditional directives e.g. @Media, @supports and @document have also selectors copied into their bodies:

```
.screen-color {
  @media screen {
    color: green;
    @media (min-width: 768px) {
      color: red;
    }
  }
  @media tv {
```

```
    color: black;
  }
}

outputs:

@media screen {
  .screen-color {
    color: green;
  }
}
@media screen and (min-width: 768px) {
  .screen-color {
    color: red;
  }
}
@media tv {
  .screen-color {
    color: black;
  }
}
```

Remaining non-conditional directives, for example font-face or keyframes, are bubbled up too. Their bodies do not change:

```
#a {
  color: blue;
  @font-face {
    src: made-up-url;
  }
  padding: 2 2 2 2;
}

outputs:

#a {
  color: blue;
}
@font-face {
  src: made-up-url;
}
#a {
  padding: 2 2 2 2;
}
```

Operations

Arithmetical operations +, -, *, / can operate on any number, color or variable. If it is possible, mathematical operations take units into account and convert numbers before

adding, subtracting or comparing them. The result has leftmost explicitly stated unit type. If the conversion is impossible or not meaningful, units are ignored. Example of impossible conversion: px to cm or rad to %.

```
// numbers are converted into the same units
@conversion-1: 5cm + 10mm; // result is 6cm
@conversion-2: 2 - 3cm - 5mm; // result is -1.5cm

// conversion is impossible
@incompatible-units: 2 + 5px - 3cm; // result is 4px

// example with variables
@base: 5%;
@filler: @base * 2; // result is 10%
@other: @base + @filler; // result is 15%
```

Multiplication and division do not convert numbers. It would not be meaningful in most cases - a length multiplied by a length gives an area and css does not support specifying areas. Less will operate on numbers as they are and assign explicitly stated unit type to the result.

```
@base: 2cm * 3mm; // result is 6cm
```

Colors are split into their red, green, blue and alpha dimensions. The operation is applied to each color dimension separately. E.g., if the user added two colors, then the green dimension of the result is equal to sum of green dimensions of input colors. If the user multiplied a color by a number, each color dimension will get multiplied.

Note: arithmetic operation on alpha is not defined, because math operation on colors do not have standard agreed upon meaning. Do not rely on current implemention as it may change in later versions.

An operation on colors always produces valid color. If some color dimension of the result ends up being bigger than ffor smaller than 00, the dimension is rounded to either ff or 00. If alpha ends up being bigger than 1.0 or smaller than 0.0, the alpha is rounded to either 1.0 or 0.0.

```
@color: #224488 / 2; //results in #112244
background-color: #112244 + #111; // result is #223355
```

Escaping

Escaping allows you to use any arbitrary string as property or variable value. Anything inside ~"anything" or ~'anything' is used as is with no changes except interpolation.

```
.weird-element {
  content: ~"^//* some horrible but needed css hack";
}
```

```
results in:
.weird-element {
  content: ^//* some horrible but needed css hack;
}
```

Functions

Less provides a variety of functions which transform colors, manipulate strings and do maths. They are documented fully in the function reference.

Using them is pretty straightforward. The following example uses percentage to convert 0.5 to 50%, increases the saturation of a base color by 5% and then sets the background color to one that is lightened by 25% and spun by 8 degrees:

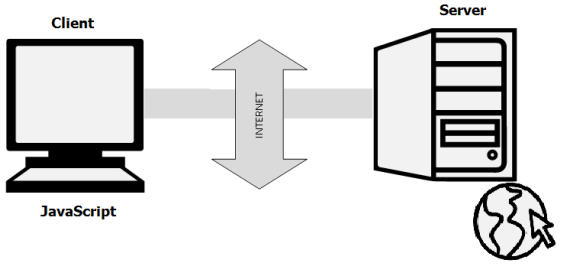
```
@base: #f04615;
@width: 0.5;
.class {
  width: percentage(@width); // returns `50%`
  color: saturate(@base, 5%);
  background-color: spin(lighten(@base, 25%), 8);
}
```

JavaScript

JavaScript is a very powerful client-side scripting language. JavaScript is used mainly for enhancing the interaction of a user with the webpage. In other words, you can make your webpage more lively and interactive, with the help of JavaScript. JavaScript is also being used widely in game development and Mobile application development.

Figure 15 JavaScript JS

JavaScript is also used in environments that are not Web-based, such as PDF documents, site-specific browsers, and desktop widgets. Newer and faster JavaScript virtual machines (VMs) and platforms built upon them have also increased the popularity of JavaScript for server-side Web applications. On the client side, developers have traditionally implemented JavaScript as an interpreted language, but more recent browsers perform just-in-time compilation. Programmers also use JavaScript in video-game development, in crafting desktop and mobile applications, and in server-side network programming with run-time environments such as Node.js.



Being a scripting language, JavaScript cannot run on its own. In fact, the browser is responsible for running JavaScript code. When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it is up to the browser to execute it. The main advantage of JavaScript is that all modern web browsers support JavaScript. So, you do not have to worry whether your site visitor uses Internet Explorer, Google Chrome, Firefox or any other browser. JavaScript will be supported. Also, JavaScript runs on any operating system including Windows, Linux or Mac. Thus, JavaScript overcomes the main disadvantages of VBScript which is limited to just IE and Windows.

Tools You Need In JavaScript: To start with, you need a text editor to write your code and a browser to display the web pages you develop. You can use text editor of your choice including Notepad++, Komodo Edit or any other text editor you are comfortable with. You can use any web browser including Internet Explorer, Google Chrome, Firefox etc.

Features

The following features are common to all conforming ECMAScript implementations, unless explicitly specified otherwise.

Universal support:

All modern Web browsers support JavaScript with built-in interpreters.

Imperative and structured:

JavaScript supports much of the structured programming syntax from C (e.g., if statements, while loops, switch statements, do while loops, etc.). One partial exception is scoping: JavaScript originally had only function scoping with var. ECMAScript 2015 added a let keyword for block scoping, meaning JavaScript now has both function and block scoping. Like C, JavaScript makes a distinction between expressions and statements. One syntactic difference from C is automatic semicolon insertion, which allows the semicolons that would normally terminate statements to be omitted.

Dynamic

Typing: As with most scripting languages, JavaScript is dynamically typed; a type is associated with each value, rather than just with each expression. For example, a variable that is at one time bound to a number may later be re-bound to a string.https://en.wikipedia.org/wiki/JavaScript JavaScript supports various ways to test the type of an object, including duck typing.

Run-time evaluation

JavaScript includes an eval function that can execute statements provided as strings at run-time.

Prototype-based (Object-oriented)

JavaScript is almost entirely object-based. In JavaScript, an object is an associative array, augmented with a prototype (see below); each string key provides the name for an object property, and there are two syntactical ways to specify such a name: dot notation (obj.x = 10) and bracket notation (obj['x'] = 10). A property may be added, rebound, or deleted at run-time. Most properties of an object (and any property that belongs to an object's prototype inheritance chain) can be enumerated using a for...inloop.

JavaScript has a small number of built-in objects, including Function and Date.

Prototypes

JavaScript uses prototypes where many other object-oriented languages use classes for inheritance. It is possible to simulate many class-based features with prototypes in JavaScript.

Functions as object constructors

Functions double as object constructors, along with their typical role. Prefixing a function call with new will create an instance of a prototype, inheriting properties and methods from the constructor (including properties from the Object prototype). ECMAScript 5 offers the Object.create method, allowing explicit creation of an instance without automatically inheriting from the Object prototype (older environments can assign the prototype to null). The constructor's prototype property determines the object used for the new object's internal prototype. New methods can be added by modifying the prototype of the function used as a constructor. JavaScript's built-in constructors, such as Array or Object, also have prototypes that can be modified. While it is possible to modify the Object prototype, it is generally considered bad practice because most objects in JavaScript will inherit methods and properties from the Object prototype, and they may not expect the prototype to be modified.

Functions as methods

Unlike many object-oriented languages, there is no distinction between a function definition and a method definition. Rather, the distinction occurs during function calling; when a function is called as a method of an object, the functions' local this keyword is bound to that object for that invocation.

Functional

A function is first-class; a function is considered to be an object. As such, a function may have properties and methods, such as .call() and .bind(). A nested function is a function defined within another function. It is created each time the outer function is invoked. In addition, each nested function forms a lexical closure: The lexical scope of the outer function (including any constant, local variable, or argument value) becomes part of the internal state of each inner function object, even after execution of the outer function concludes. JavaScript also supports anonymous functions.

Delegative

JavaScript supports implicit and explicit delegation.

Functions as roles (Traits and Mixins)

JavaScript natively supports various function-based implementations of Role patterns like Traits and Mixins. Such a function defines additional behavior by at least one method bound to the this keyword within its function body. A Role then has to be delegated explicitly via call or apply to objects that need to feature additional behavior that is not shared via the prototype chain.

Object composition and inheritance

Whereas explicit function-based delegation does cover composition in JavaScript, implicit delegation already happens every time the prototype chain is walked in order to, e.g., find a method that might be related to but is not directly owned by an object. Once the method is found it gets called within this object's context. Thus inheritance in JavaScript is covered by a delegation automatism that is bound to the prototype property of constructor functions.

Miscellaneous

Run-time environment: JavaScript typically relies on a run-time environment (e.g., a Web browser) to provide objects and methods by which scripts can interact with the environment (e.g., a webpage DOM). It also relies on the run-time environment to provide the ability to include/import scripts (e.g., HTML <script> elements). This is not a language feature per se, but it is common in most JavaScript implementations.

JavaScript processes messages from a queue one at a time. Upon loading a new message, JavaScript calls a function associated with that message, which creates a call stack frame (the function's arguments and local variables). The call stack shrinks and grows based on the function's needs. Upon function completion, when the stack is empty, JavaScript proceeds to the next message in the queue. This is called the event loop, described as "run to completion" because each message is fully processed before the next message is considered. However, the language's concurrency model describes the event loop as non-blocking: program input/output is performed using events and callback functions. This means, for instance, that JavaScript can process a mouse click while waiting for a database query to return information.

Variadic functions

An indefinite number of parameters can be passed to a function. The function can access them through formal parameters and also through the local arguments object. Variadic functions can also be created by using the bind method.

Array and object literals

Like many scripting languages, arrays and objects (associative arrays in other languages) can each be created with a succinct shortcut syntax. In fact, these literals form the basis of the JSON data format.

Regular expressions

JavaScript also supports regular expressions in a manner similar to Perl, which provide a concise and powerful syntax for text manipulation that is more sophisticated than the built-in string functions.

Vendor-specific extensions

JavaScript is officially managed by Mozilla Foundation, and new language features are added periodically. However, only some JavaScript engines support these new features:

- property getter and setter functions (supported by WebKit, Gecko, Opera, ActionScript, and Rhino)
- conditional catch clauses
- iterator protocol (adopted from Python)
- shallow generators-coroutines (adopted from Python)
- array comprehensions and generator expressions (adopted from Python)
- proper block scope via the let keyword
- array and object destructuring (limited form of pattern matching)
- concise function expressions (function(args) expr)
- ECMAScript for XML (E4X), an extension that adds native XML support to ECMAScript (unsupported in Firefox since version 21)

Syntax:

Simple examples

Variables in JavaScript can be defined using the var keyword:

```
var x; // defines the variable x and assigns to it the special value "undefined"
(not to be confused with an undefined value)
var y = 2; // defines the variable y and assigns to it the value 2
```

Note the comments in the example above, both of which were preceded with two forward slashes.

There is no built-in I/O functionality in JavaScript; the run-time environment provides that. The ECMAScript specification in edition 5.1 mentions:

Indeed, there are no provisions in this specification for input of external data or output of computed results.

However, most runtime environments have a console object that can be used to print output. Here is a minimalist Hello World program in JavaScript:

```
console.log("Hello World!");
```

A simple recursive function:

```
function factorial(n) {
   if (n === 0 || n === 1) {
      return 1;  // 0! = 1! = 1
   }
   return n * factorial(n - 1);
}
factorial(3); // returns 6
```

An anonymous function (or lambda):

```
function counter() {
   var count = 0;
   return function() {
      return ++count;
   };
}
var closure = counter();
closure(); // returns 1
closure(); // returns 2
closure(); // returns 3
function sum() { This example shows that in JavaScript, function
closures captures their non-local variables by reference.
```

Variadic function demonstration (arguments is a special variable):

```
   var x = 0;
```

```
    for (var i = 0; i < arguments.length; ++i) {
        x += arguments[i];
    }
    return x;
}
sum(1, 2); // returns 3
sum(1, 2, 3); // returns 6
```

Immediately-invoked function expressions are often used to create modules, as before ECMAScript 2015 there was not built-in construct in the language. Modules allow gathering properties and methods in a namespace and making some of them private:

```
var counter = (function () {
    var i = 0; // private property
    return {   // public methods
        get: function () {
            alert(i);
        },
        set: function (value) {
            i = value;
        },
        increment: function () {
            alert(++i);
        }
    };
})(); // module
counter.get();      // shows 0
counter.set(6);
counter.increment(); // shows 7
counter.increment(); // shows 8
```

jQuery

jQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. It is free, open-source software using the permissive MIT license. Web analysis indicates that it is the most widely deployed JavaScript library by a large margin.

jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets. The modular approach to the jQuery library allows the creation of powerful dynamic web pages and Web applications.

Figure 16 jQuery

The set of jQuery core features—DOM element selections, traversal and manipulation—enabled by its selector engine (named "Sizzle" from v1.3), created a new "programming style", fusing algorithms and DOM data structures. This style influenced the architecture of other JavaScript frameworks like YUI v3 and Dojo, later stimulating the creation of the standard Selectors API.

Microsoft and Nokia bundle jQuery on their platforms. Microsoft includes it with Visual Studio for use within Microsoft's ASP.NET AJAX and ASP.NET MVC frameworks while Nokia has integrated it into the Web Run-Time widget development platform.

jQuery, at its core, is a Document Object Model (DOM) manipulation library. The DOM is a tree-structure representation of all the elements of a Web page. jQuery simplifies the syntax for finding, selecting, and manipulating these DOM elements. For example, jQuery can be used for finding an element in the document with a certain property (e.g. all elements with an h1 tag), changing one or more of its attributes (e.g. color, visibility), or making it respond to an event (e.g. a mouse click).

jQuery also provides a paradigm for event handling that goes beyond basic DOM element selection and manipulation. The event assignment and the event callback function definition are done in a single step in a single location in the code. jQuery also aims to incorporate other highly used JavaScript functionality (e.g. fade ins and fade outs when hiding elements, animations by manipulating CSS properties).

The principles of developing with jQuery are:

- **Separation of JavaScript and HTML**: The jQuery library provides simple syntax for adding event handlers to the DOM using JavaScript, rather than adding HTML event attributes to call JavaScript functions. Thus, it encourages developers to completely separate JavaScript code from HTML markup.
- **Brevity and clarity**: jQuery promotes brevity and clarity with features like chainable functions and shorthand function names.
- **Elimination of cross-browser incompatibilities**: The JavaScript engines of different browsers differ slightly so JavaScript code that works for one browser may not work for another. Like other JavaScript toolkits, jQuery handles all these cross-browser inconsistencies and provides a consistent interface that works across different browsers.
- **Extensibility**: New events, elements, and methods can be easily added and then reused as a plugin.

Features

- DOM element selections using the multi-browser open source selector engine Sizzle, a spin-off of the jQuery project
- DOM manipulation based on CSS selectors that uses elements' names and attributes, such as id and class, as criteria to select nodes in the DOM
- Events
- Effects and animations
- Ajax
- Deferred and Promise objects to control asynchronous processing
- JSON parsing
- Extensibility through plug-ins
- Utilities, such as feature detection

- Compatibility methods that are natively available in modern browsers, but need fall backs for older ones, such as inArray() and each()
- Multi-browser (not to be confused with cross-browser) support

MySQL

MySQL is a fast, easy-to-use RDBMS (Relation Database Management System) being used for many small and big businesses. MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company. MySQL is written in C and C++. Its SQL parser is written in yacc, but it uses a home-brewed lexical analyzer MySQL works on many system platforms, including AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, macOS, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos and Tru64. A port of MySQL to OpenVMS also exists.

Figure 17 My SQL

The MySQL server software itself and the client libraries use dual-licensing distribution. They are offered under GPL version 2, beginning from 28 June 2000https://en.wikipedia.org/wiki/MySQL (which in 2009 has been extended with a FLOSS License Exception) or to use a proprietary license.

Support can be obtained from the official manual. Free support additionally is available in different IRC channels and forums. Oracle offers paid support via its MySQL Enterprise products. They differ in the scope of services and in price. Additionally, a number of third party organisations exist to provide support and services, including MariaDB and Percona.

MySQL has received positive reviews, and reviewers noticed it "performs extremely well in the average case" and that the "developer interfaces are there, and the documentation (not to mention feedback in the real world via Web sites and the like) is very, very good". It has also been tested to be a "fast, stable and true multi-user, multi-threaded sql database server". MySQL is becoming so popular because of many good reasons:

MySQL is released under an open-source license. So you have nothing to pay to use it.

MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.

MySQL uses a standard form of the well-known SQL data language.

MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.

MySQL works very quickly and works well even with large data sets.

MySQL is very friendly to PHP, the most appreciated language for web development.

MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).

MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

Features

MySQL is offered under two different editions: the open source MySQL Community Server and the proprietary Enterprise Server. MySQL Enterprise Server is differentiated by a series of proprietary extensions which install as server plugins, but otherwise shares the version numbering system and is built from the same code base.

Major features as available in MySQL 5.6

- A broad subset of ANSI SQL 99, as well as extensions
- Cross-platform support
- Stored procedures, using a procedural language that closely adheres to SQL/PSM
- Triggers
- Cursors
- Updatable views
- Online DDL when using the InnoDB Storage Engine.
- Information schema
- Performance Schema that collects and aggregates statistics about server execution and query performance for monitoring purposes.
- A set of SQL Mode options to control runtime behavior, including a strict mode to better adhere to SQL standards.
- X/Open XA distributed transaction processing (DTP) support; two phase commit as part of this, using the default InnoDB storage engine
- Transactions with savepoints when using the default InnoDB Storage Engine. The NDB Cluster Storage Engine also supports transactions.
- ACID compliance when using InnoDB and NDB Cluster Storage Engines
- SSL support
- Query caching
- Sub-SELECTs (i.e. nested SELECTs)
- Built-in replication support (i.e., master-master replication and master-slave replication) with one master per slave, many slaves per master. Multi-master replication is provided in MySQL Cluster, and multi-master support can be added to unclustered configurations using Galera Cluster.
- Full-text indexing and searching
- Embedded database library
- Unicode support
- Partitioned tables with pruning of partitions in optimizer
- Shared-nothing clustering through MySQL Cluster
- Multiple storage engines, allowing one to choose the one that is most effective for each table in the application.
- Native storage engines InnoDB, MyISAM, Merge, Memory (heap), Federated, Archive, CSV, Blackhole, NDB Cluster.
- Commit grouping, gathering multiple transactions from multiple connections together to increase the number of commits per second.

The developers release minor updates of the MySQL Server approximately every two months. The sources can be obtained from MySQL's website or from MySQL's GitHub repository, both under the GPL license.

User Interfaces

Graphical user interfaces

A graphical user interface (GUI) is a type of interface that allows users to interact with electronic devices or programs through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation. GUIs are easier to learn than command-line interfaces (CLIs), which require commands to be typed on the keyboard.

Third-party proprietary and free graphical administration applications (or "front ends") are available that integrate with MySQL and enable users to work with database structure and data visually. Some well-known front ends are:

Figure 18 MySQL Workbench running on macOS

MySQL Workbench

MySQL Workbench is the official integrated environment for MySQL. It was developed by MySQL AB, and enables users to graphically administer MySQL databases and visually design database structures. MySQL Workbench replaces the previous package of software, MySQL GUI Tools. Similar to other third-party packages, but still considered the authoritative MySQL front end, MySQL Workbench lets users manage database design & modelling, SQL development (replacing MySQL Query Browser) and Database administration (replacing MySQL Administrator).

MySQL Workbench is available in two editions, the regular free and open source Community Edition which may be downloaded from the MySQL website, and the proprietary Standard Edition which extends and improves the feature set of the Community Edition.

Adminer

Adminer (formerly known as phpMinAdmin) is a free MySQL front end for managing content in MySQL databases (since version 2, it also works on PostgreSQL, MS SQL, SQLite and Oracle SQL databases). Adminer is distributed under the Apache license (or GPL v2) in the form of a single PHP file (around 300 KiB in size), and is capable of managing multiple databases, with many CSS skins available. Its author is Jakub Vrána who started to develop this tool as a light-weight alternative to phpMyAdmin, in July 2007.

Database Workbench

Database Workbench is a software application for development and administration of multiple relational databases using SQL, with interoperationality between different database systems, developed by Upscene Productions.

Because Databases Workbench supports multiple database systems, it can provide software developers with the same interface and development environment for these otherwise different database systems and also includes cross database tools.

Database Workbench supports the following relational databases: Oracle Database, Microsoft SQL Server, SQL Anywhere, Firebird, NexusDB, InterBase, MySQL and MariaDB. Database Workbench 5 runs on 32-bit or 64 bit Windows platforms. Under Linux, FreeBSD or macOS Database Workbench can operate using Wine.

DBEdit

DBEdit is a database editor, which can connect to an Oracle, DB2, MySQL and any database that provides a JDBC driver. It runs on Windows, Linux and Solaris. DBEdit is free and open source software and distributed under the GNU General Public License. The source code is hosted on SourceForge.

HeidiSQL

HeidiSQL, previously known as MySQL-Front, is a free and open source client, or frontend for MySQL (and for its forks like MariaDB and Percona Server), Microsoft SQL Server and PostgreSQL. HeidiSQL is developed by German programmer Ansgar Becker and a few other contributors in Delphi. To manage databases with HeidiSQL, users must login to a local or remote MySQL server with acceptable credentials, creating a session. Within this session users may manage MySQL Databases within the connected MySQL server, disconnecting from the server when done. Its feature set is sufficient for most common and advanced database, table and data record operations but remains in active development to move towards the full functionality expected in a MySQL Frontend.

LibreOffice Base

LibreOffice Base allows the creation and management of databases, preparation of forms and reports that provide end users easy access to data. Like Microsoft Access, it can be used as a front-end for various database systems, including Access databases (JET), ODBC data sources, and MySQL or PostgreSQL

Navicat

Navicat is a series of graphical database management and development software produced by PremiumSoft CyberTech Ltd. for MySQL, MariaDB, Oracle, SQLite, PostgreSQL and Microsoft SQL Server. It has an Explorer-like graphical user interface and supports multiple database connections for local and remote databases. Its design is made to meet the needs of a variety of audiences, from database administrators and programmers to various businesses/companies that serve clients and share information with partners.

Navicat is a cross-platform tool and works on Microsoft Windows, OS X and Linux platforms. Upon purchase, users are able to select a language for the software from eight available languages: English, French, German, Spanish, Japanese, Polish, Simplified Chinese and Traditional Chinese.

OpenOffice.org

OpenOffice.org Base is freely available and can manage MySQL databases if the entire suite is installed.

PhpMyAdmin

phpMyAdmin is a free and open source tool written in PHP intended to handle the administration of MySQL with the use of a web browser. It can perform various tasks such as creating, modifying or deleting databases, tables, fields or rows; executing SQL statements; or managing users and permissions. The software, which is available in 78 languages, is maintained by The phpMyAdmin Project.

It can import data from CSV and SQL, and transform stored data into any format using a set of predefined functions, like displaying BLOB-data as images or download-links.

SQLBuddy

SQLBuddy is an open-source web-based application written in PHP intended to handle the administration of MySQL and SQLite with the use of a Web browser. The project places an emphasis on ease of installation and a simple user interface.

SQLyog

SQLyog is a GUI tool available in free as well as paid versions. Data manipulations (e.g., insert, update, and delete) may be done from a spreadsheet-like interface. Its editor has syntax highlighting and various automatic formatting options. Both raw table data and a result set from a query can be manipulated. Its data search feature uses Google-like search syntax and translates to SQL transparently for the user. It has a backup tool for performing unattended backups. Backups may be compressed and optionally stored as a file-per-table as well as identified with a timestamp.

Toad for MySQL

Toad for MySQL is a software application from Dell Software that database developers, database administrators and data analysts use to manage both relational and non-relational databases using SQL. Toad supports many databases and environments. It runs on all 32-bit/64-bit Windows platforms, including Microsoft Windows Server, Windows XP, Windows Vista, Windows 7 and 8 (32-Bit or 64-Bit). Dell Software has also released a Toad Mac Edition. Dell provides Toad in commercial and trial/freeware versions. The freeware version is available from the ToadWorld.com community.

Webmin

Webmin is a web-based system configuration tool for Unix-like systems, although recent versions can also be installed and run on Windows. With it, it is possible to configure operating system internals, such as users, disk quotas, services or configuration files, as well as modify and control open source apps, such as the Apache HTTP Server, PHP or MySQL.

Webmin is largely based on Perl, running as its own process and web server. It defaults to TCP port 10000 for communicating, and can be configured to use SSL if OpenSSL is installed with additional required Perl Modules.

It is built around modules, which have an interface to the configuration files and the Webmin server. This makes it easy to add new functionality. Due to Webmin's modular design, it is possible for anyone who is interested to write plugins for desktop configuration.

Webmin also allows for controlling many machines through a single interface, or seamless login on other webmin hosts on the same subnet or LAN.

Command-line interfaces

A command-line interface is a means of interacting with a computer program where the user issues commands to the program by typing in successive lines of text (command lines). MySQL ships with many command line tools, from which the main interface is the mysql client.

MySQL Utilities is a set of utilities designed to perform common maintenance and administrative tasks. Originally included as part of the MySQL Workbench, the utilities are a stand-alone download available from Oracle.

Percona Toolkit is a cross-platform toolkit for MySQL, developed in Perl. Percona Toolkit can be used to prove replication is working correctly, fix corrupted data, automate repetitive tasks, and speed up servers. Percona Toolkit is included with several Linux distributions such as CentOS and Debian, and packages are available for Fedora and Ubuntu as well. Percona Toolkit was originally developed as Maatkit, but as of late 2011, Maatkit is no longer developed.

Hardware Interfaces

- **Compatible OS**: Windows XP SP3+, Windows Vista SP2+, Windows Server 2003 SP2+, Windows 7, Windows Server 2008, Windows Server 2008 R2.
    - **Note*: You must have administrator privileges on your computer to run AMPPS.
- **Space**: Capacity of minimum 1.5GB Hard Disk space.
- **Memory**: 1GB RAM

Above are the bare minimum requirements for the WAMP stack we will be using on the server side to run Apache, PHP and MySQL.

Data Models

DFD

Context Level Data Flow Diagram

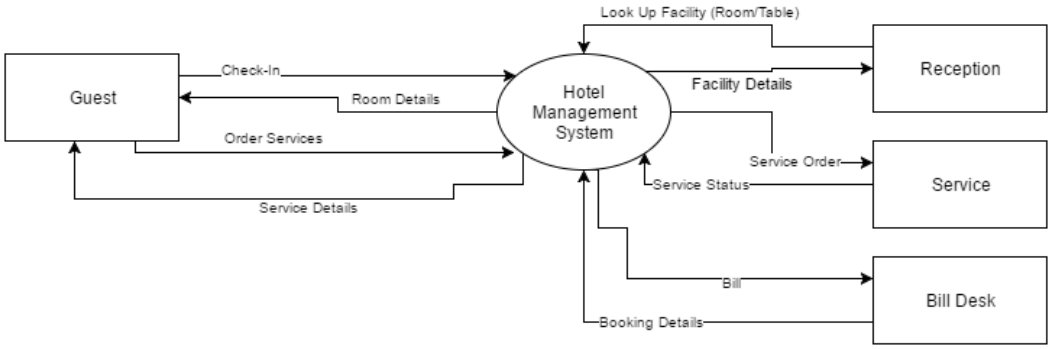


Figure 19 Context DFD

Data Flow while generating Bill

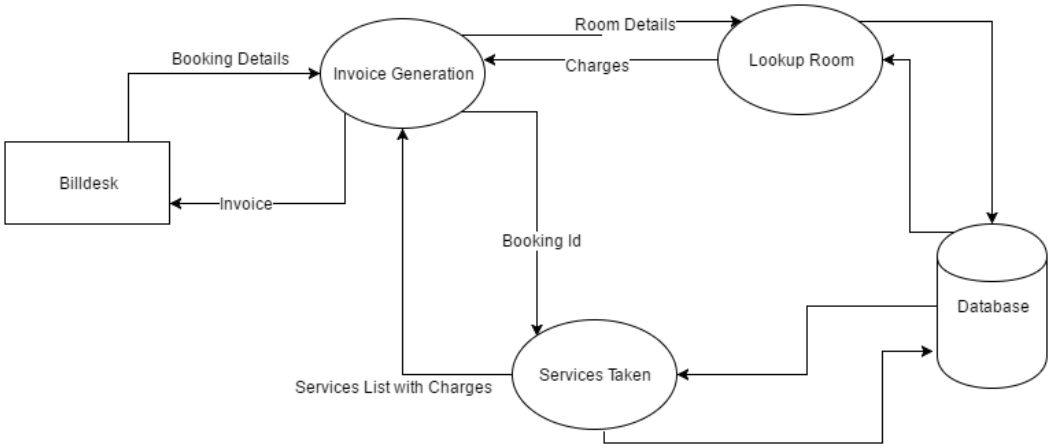


Figure 20 DFD Bill Creation

Check-In

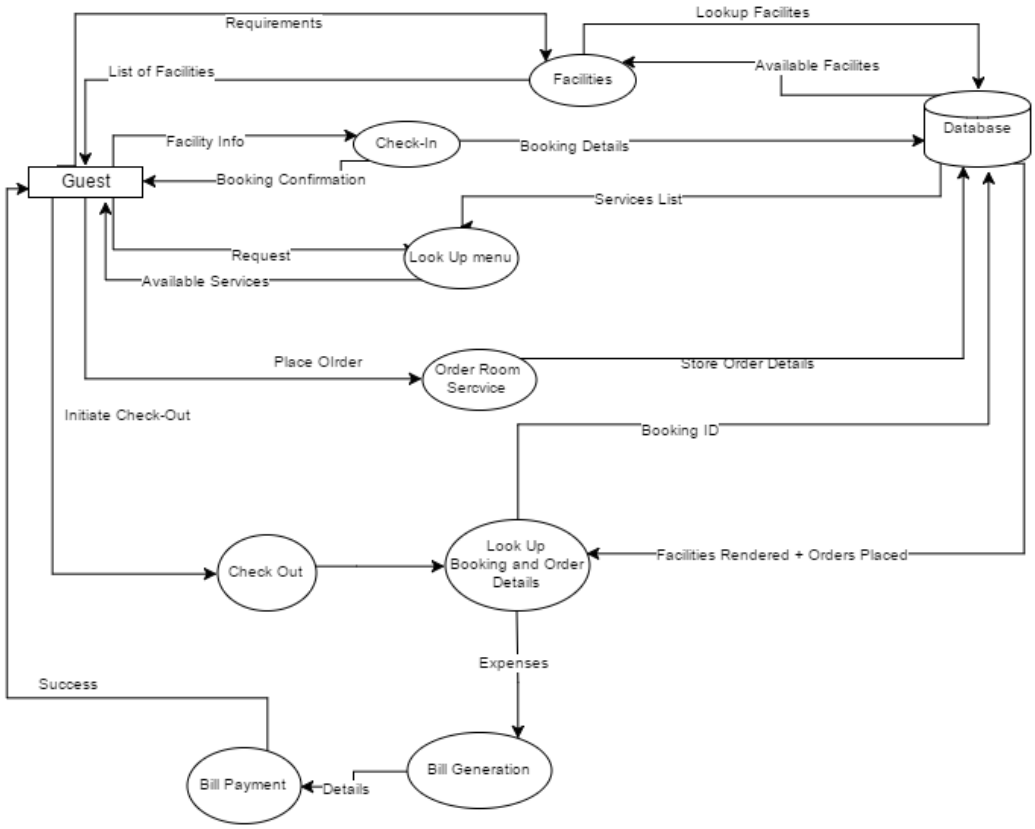


Figure 21 DFD Check-In

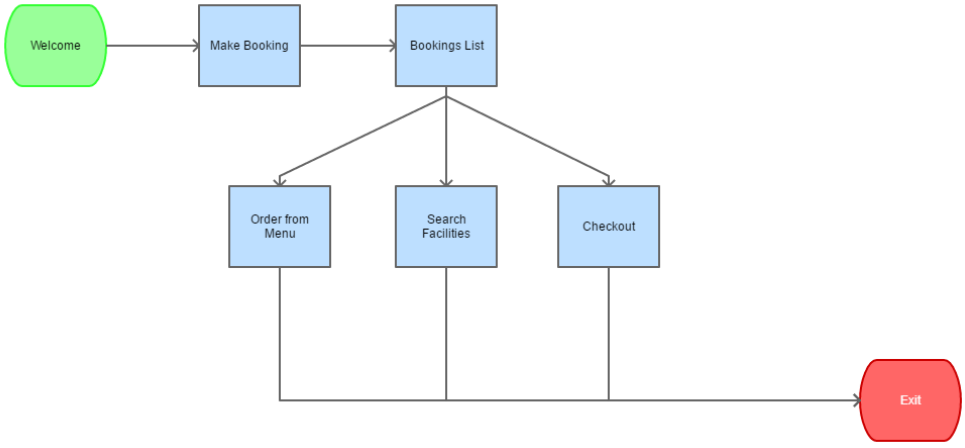


Figure 22 Base

Class Diagrams

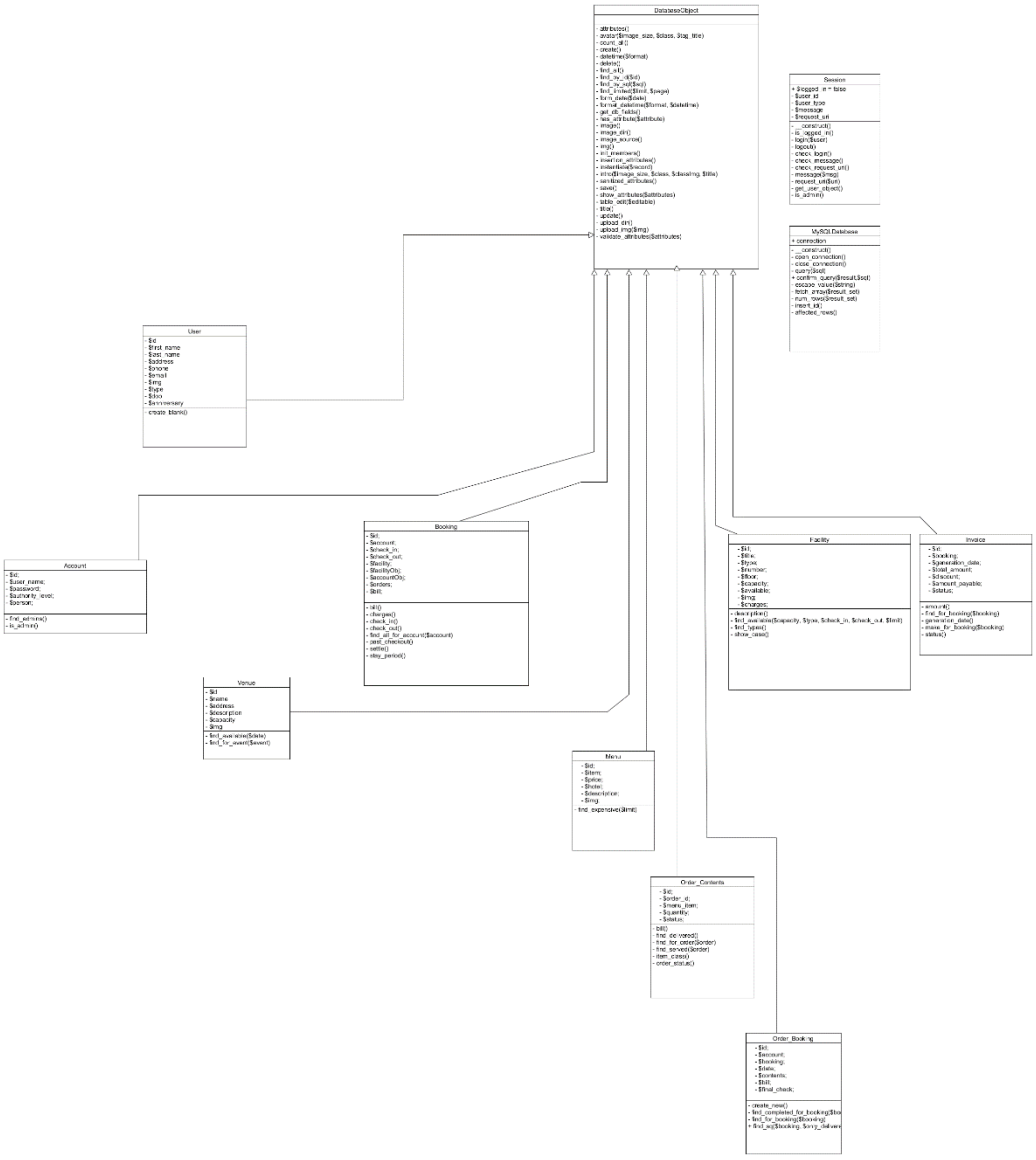


Figure 23 Class Diagram

Use Case Diagrams

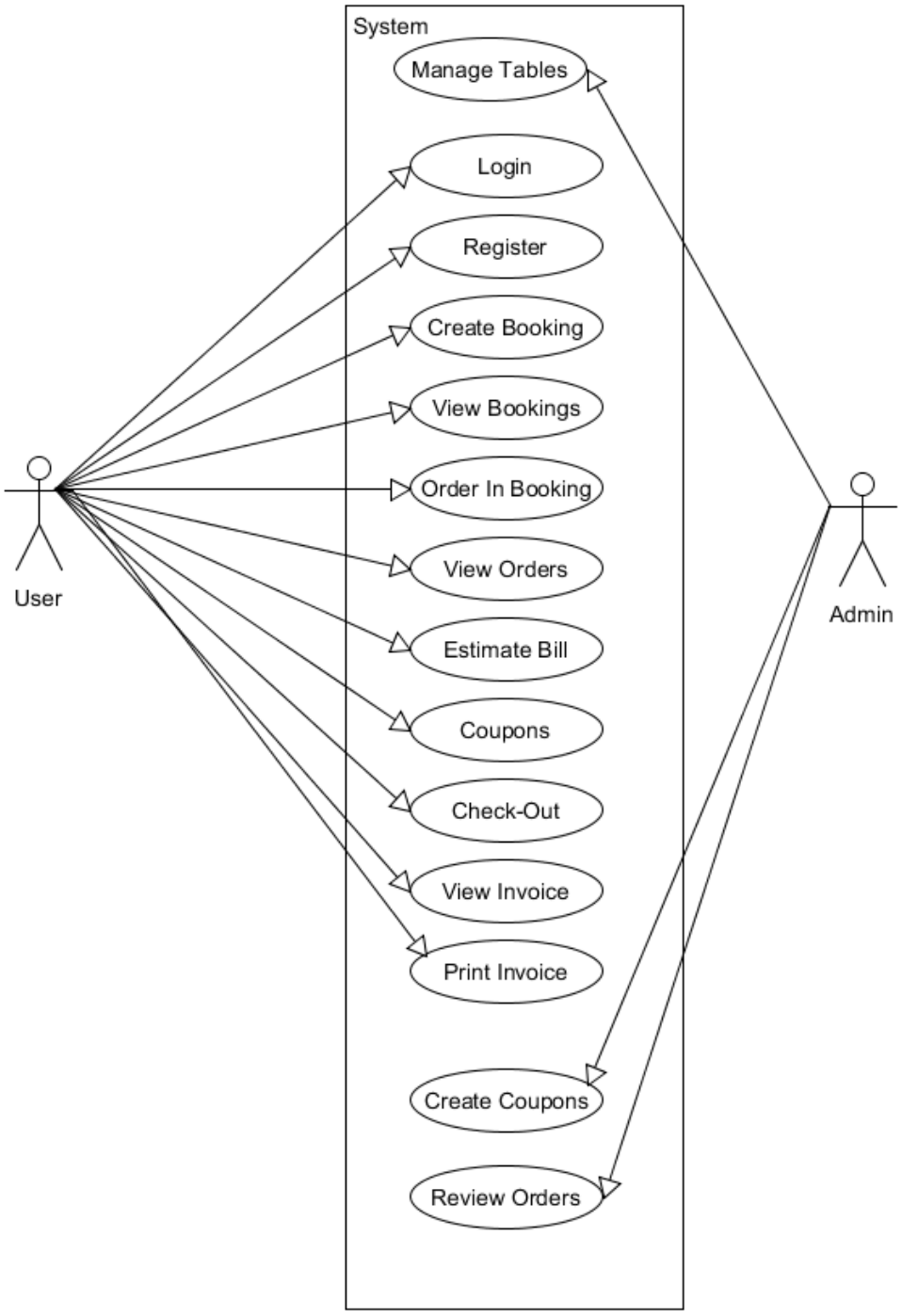


Figure 24 Use Case

System Design

Modularisation Details

Admin Module

There will be tasks in the project that require supervision. Such as to validate if the room is accurately booked, or in fact available in the list. Same goes for user management tasks such as forgotten passwords, etc. So the admin module will cater to those needs and will have such tasks that only admins can perform. These admin account will be created by other admins or the super admin whose account is of the owner of the project.

But the major portion of the application will be from the perspective of the common user as an admin can also be required to perform such tasks.

Tables Management

This will allow admins to view all the database tables there are in the system and make sure that everything is in check. Also this same portal helps an admin in creating / editing / deleting.

- Users
- Facilities
- Orders
- Menu Items
- Invoice Discounts
- So on…

User Modules

Following are the modules that **every user** of the project can operate onto. These are to be created in a highly cohesive and loosely coupled manner to ensure further applications and enhancements in later stages.

Booking of Facility

This will allow the users to create a new booking with filling out a simple form that asks them to input the Personal details if they do not have account yet, type of facility, date and time of check in and check out.

Managing a Booking

Any user who has booked a facility can view the details regarding the booking anytime they wish to. Management of a Booking includes

- Ordering Items
  - Food
  - Room Service
  - Etc
- Confirming purchases
- Checkout anytime
- Pay Bill

Hotel Staff Tasks

Booking Management

Staff can see all the requests of facilities made by all the guests and manage their status.

Adding New Services

An authorised person can create new services that are available in the hotel so the guests are more and more pleased always.

Database Design

Database design is the process of producing a detailed data model of database. This data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views. In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system (DBMS).

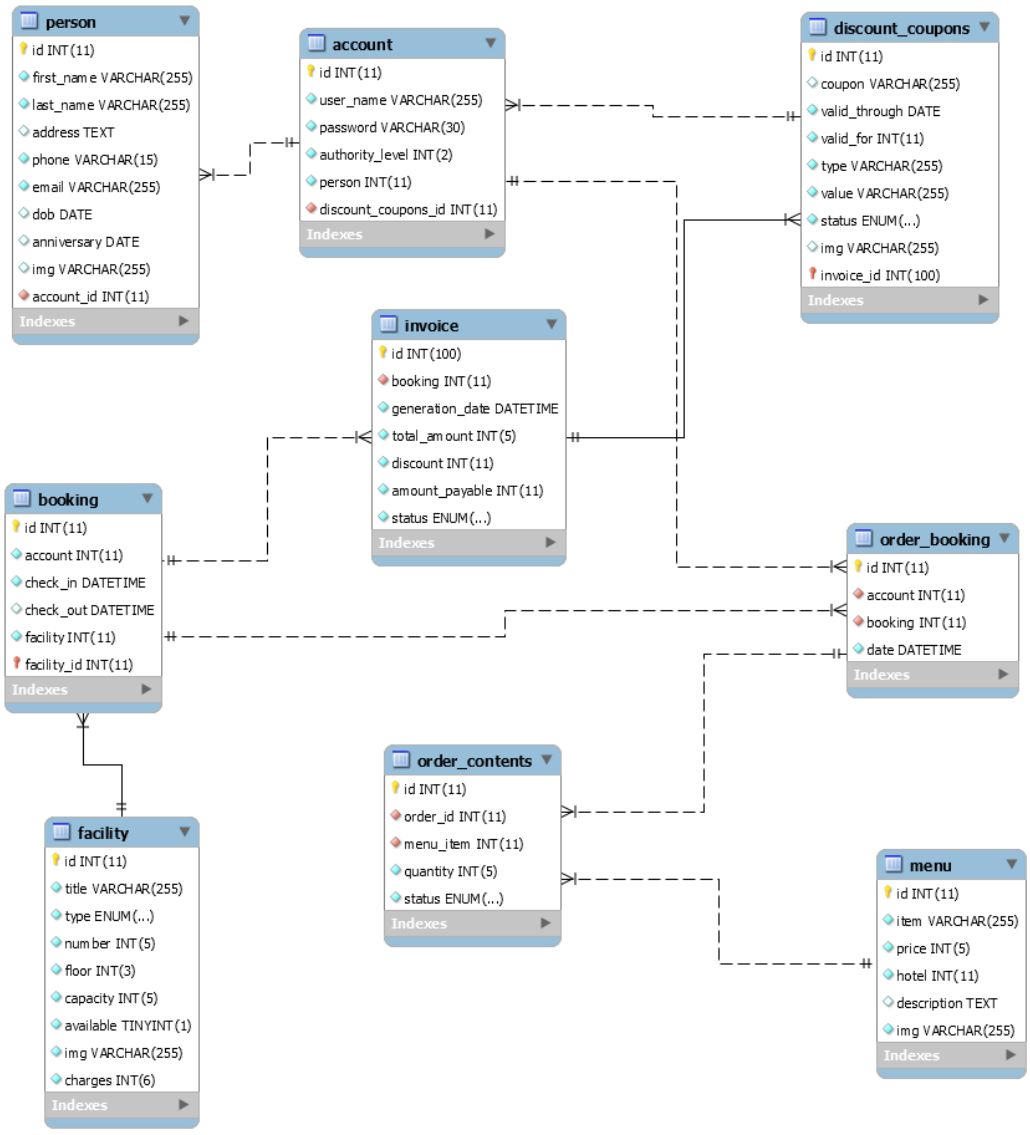
The process of doing database design generally consists of a number of steps which will be carried out by the database designer. Usually, the designer must:

Determine the data to be stored in the database.

Determine the relationships between the different data elements.

Superimpose a logical structure upon the data on the basis of these relationships.

Within the relational model the final step above can generally be broken down into two further steps that of determining the grouping of information within the system, generally determining what are the basic objects about which information is being stored, and then determining the relationships between these groups of information, or objects. This step is not necessary with an Object database.



a0073

Figure 25 ER Diagram

User Interface Design

User interface design (UI) or user interface engineering is the design of user interfaces for machines and software, such as computers, home appliances, mobile devices, and other electronic devices, with the focus on maximizing usability and the user experience. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals (user centric design).

Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to itself. Graphic design and typography are utilized to support its usability, influencing how the user performs certain interactions and improving the aesthetic appeal of the design; design aesthetics may enhance or detract from the ability of users to use the functions of the interface. The design process must balance technical

functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

Interface design is involved in a wide range of projects from computer systems, to cars, to commercial planes; all of these projects involve much of the same basic human interactions yet also require some unique skills and knowledge. As a result, designers tend to specialize in certain types of projects and have skills focused on their expertise, whether that be software design, user research, web design, or industrial design.

Test Cases

A test case is a document, which has a set of test data, preconditions, expected results and postconditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution postcondition.

Benefits of Test Driven Design Approach

- Much less debug time
- Code proven to meet requirements
- Tests become Safety Net
- Near zero defects
- Shorter development cycles

Basic Tests conducted during the Project

Test Scenario	**Test Description**	**Input**	**Expected Outcome**
User Login	Upon Input of Correct Username Password combination the user should be allowed to enter the system	Username Password	User Page for Logged in account
User Registeration	First time users fill out a simple form to gain access into the system	Personal Details Username Password	User Page for newly created Account
Booking Creation	Fill out Boooking details and check in	Check IN Check OUT Persons and Type of Facility	Booking Page
Order	Select Items from Menu and order them	Menu Items list	Order Placed
Checkout	Create Estimated bill for booking	Booking ID	Estimate amount of orders
Invoice	Generate invoice upon checkout	Booking Details	Task assignment to correct member and displayed in their portal

Image Upload	Upload an Image to event's gallery	Image	Image displayed in event's gallery
Image Comment	Comment on an image	Comment text	Comment shown in the comments panel of Image viewer under User's name who posted the comment
Logout	Exit the portal by ending session	Null	Secure pages require login again
Add Schedule	Select an event and add a schedule to it	New schedule's time	Schedule added in the event.

Coding

SQL commands

Table structure for table `person`

```
CREATE TABLE `person` (
  `id` int(11) UNSIGNED NOT NULL,
  `first_name` varchar(255) NOT NULL,
  `last_name` varchar(255) NOT NULL,
  `address` text,
  `phone` varchar(15) NOT NULL,
  `email` varchar(255) NOT NULL,
  `dob` date DEFAULT NULL,
  `anniversary` date DEFAULT NULL,
  `img` varchar(255) DEFAULT 'default.jpg'
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Dumping data for table `person`

```
INSERT INTO `person` (`id`, `first_name`, `last_name`, `address`, `phone`,
`email`, `dob`, `anniversary`, `img`) VALUES
(1, 'Mandeep', 'Kaur', '65 Jalandhar', '00002', 'mandeepshabina@gmail.com',
'1993-07-27', '1993-07-27', '1490066978-2223-p1.jpg'),
(2, 'Lakshay', 'Verma', '8, 14 Jalandhar Cantt', '9779333346',
'verma_lakshay@live.in', '1993-10-31', '2011-03-25', '1490502297-2903-
lk.jpg'),
(3, 'Vipul', 'Gupta', 'Adarsh Nagar Jalandhar', '123485', 'vipulgupta@gmail.com',
'1993-10-15', '2017-03-05', '1490142904-9391-g1.jpg'),
(4, 'Tarun', 'Veer', 'BASANT AVENUE', '5555', 'tarun@gmail.com', '1993-07-07',
'2017-03-03', '1490142885-1297-images.png'),
(5, 'Rohit', 'Kural', 'deep nagar', '21', 'rohit@gmail.com', '1990-04-29', '2015-
01-19', '1490142930-6673-g3.jpg');
```

Table structure for table `account`

```
CREATE TABLE `account` (
  `id` int(11) UNSIGNED NOT NULL,
  `user_name` varchar(255) NOT NULL,
  `password` varchar(30) NOT NULL,
  `authority_level` int(2) NOT NULL,
  `person` int(11) UNSIGNED NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Dumping data for table `account`

```
INSERT INTO `account` (`id`, `user_name`, `password`, `authority_level`,
`person`) VALUES
(1, 'mandeep', 'shabina', 3, 1),
(2, 'lakshay', '8946553', 3, 2),
(3, 'mandyk', '123456', 1, 1),
(4, 'vipul1', '123456', 1, 3),
(5, ‘rohit’, 'supass', 1, 0);
```

Table structure for table `facility`

```
CREATE TABLE `facility` (
  `id` int(11) UNSIGNED NOT NULL,
  `title` varchar(255) NOT NULL,
  `type` enum('Deluxe Suite','Diamond Room','Golden Room','Silver
Room','Hall','Table') NOT NULL DEFAULT 'Silver Room',
  `number` int(5) NOT NULL,
  `floor` int(3) NOT NULL,
  `capacity` int(5) NOT NULL,
  `available` tinyint(1) NOT NULL,
  `img` varchar(255) NOT NULL DEFAULT 'default.jpg',
  `charges` int(6) UNSIGNED NOT NULL DEFAULT '12000'
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Dumping data for table `facility`

```
INSERT INTO `facility` (`id`, `title`, `type`, `number`, `floor`, `capacity`,
`available`, `img`, `charges`) VALUES
(1, 'Deluxe', 'Hall', 2, 5, 50, 1, '1490062697-5629-c1.jpg', 12000),
(2, 'Beach Side', 'Golden Room', 1, 1, 5, 1, '1490062720-8617-c2.jpg', 12000),
(3, 'Pool side', 'Silver Room', 1, 1, 2, 1, '1490062753-6854-c3.jpg', 12000),
(4, 'Cummings', 'Diamond Room', 30, 2, 2, 0, 'default.jpg', 12000),
(5, 'Pitts', 'Hall', 63, 4, 50, 0, 'default.jpg', 12000),
(6, 'Santiago', 'Golden Room', 59, 6, 2, 0, 'default.jpg', 12000),
(7, 'Valenzuela', 'Table', 25, 1, 5, 0, 'default.jpg', 12000),
(8, 'Conner', 'Diamond Room', 58, 6, 2, 1, 'default.jpg', 12000),
(9, 'Irwin', 'Golden Room', 57, 1, 2, 1, 'default.jpg', 12000),
(10, 'Wright', 'Table', 25, 4, 10, 0, 'default.jpg', 12000),
(11, 'Garrison', 'Silver Room', 21, 5, 2, 1, 'default.jpg', 12000),
(12, 'Miranda', 'Golden Room', 40, 4, 2, 1, 'default.jpg', 12000),
(13, 'Porter', 'Silver Room', 48, 1, 2, 1, 'default.jpg', 12000),
(14, 'Mclaughlin', 'Table', 67, 5, 5, 1, 'default.jpg', 12000),
(15, 'Palmer', 'Golden Room', 88, 4, 2, 0, 'default.jpg', 12000),
(16, 'Dunlap', 'Diamond Room', 75, 2, 2, 0, 'default.jpg', 12000),
(17, 'Rutledge', 'Silver Room', 88, 1, 2, 0, 'default.jpg', 12000),
(18, 'Ortega', 'Golden Room', 53, 1, 2, 0, 'default.jpg', 12000),
```

(19, 'Howe', 'Silver Room', 48, 6, 2, 0, 'default.jpg', 12000),
(20, 'Strong', 'Table', 24, 4, 10, 0, 'default.jpg', 12000),
(21, 'Mcfadden', 'Golden Room', 67, 5, 2, 1, 'default.jpg', 12000),
(22, 'Forbes', 'Silver Room', 28, 2, 4, 1, 'default.jpg', 12000),
(23, 'Whitehead', 'Silver Room', 20, 1, 2, 1, 'default.jpg', 12000),
(24, 'Coffey', 'Golden Room', 69, 6, 2, 1, 'default.jpg', 12000),
(25, 'Griffin', 'Hall', 30, 3, 50, 1, 'default.jpg', 12000),
(26, 'Perry', 'Silver Room', 27, 6, 4, 0, 'default.jpg', 12000),
(27, 'Beck', 'Golden Room', 9, 3, 2, 1, 'default.jpg', 12000),
(28, 'Salinas', 'Diamond Room', 90, 4, 2, 1, 'default.jpg', 12000),
(29, 'Hurley', 'Silver Room', 42, 4, 2, 0, 'default.jpg', 12000),
(30, 'Mccall', 'Table', 44, 1, 10, 1, 'default.jpg', 12000),
(31, 'Herring', 'Silver Room', 98, 4, 2, 1, 'default.jpg', 12000),
(32, 'Campbell', 'Diamond Room', 18, 6, 2, 0, 'default.jpg', 12000),
(33, 'Barron', 'Golden Room', 50, 6, 2, 0, 'default.jpg', 12000),
(34, 'Kane', 'Silver Room', 96, 6, 4, 0, 'default.jpg', 12000),
(35, 'Glenn', 'Hall', 66, 2, 50, 0, 'default.jpg', 12000),
(36, 'Peck', 'Golden Room', 94, 2, 2, 0, 'default.jpg', 12000),
(37, 'Rutledge', 'Silver Room', 60, 5, 2, 1, 'default.jpg', 12000),
(38, 'Collier', 'Silver Room', 25, 1, 4, 1, 'default.jpg', 12000),
(39, 'Medina', 'Golden Room', 30, 2, 2, 1, 'default.jpg', 12000),
(40, 'Ortega', 'Table', 84, 1, 10, 1, 'default.jpg', 12000),
(41, 'Pruitt', 'Silver Room', 93, 1, 2, 1, 'default.jpg', 12000),
(42, 'Scott', 'Golden Room', 40, 1, 2, 1, 'default.jpg', 12000),
(43, 'Rasmussen', 'Silver Room', 48, 2, 2, 0, 'default.jpg', 12000),
(44, 'Duncan', 'Hall', 46, 1, 153, 0, 'default.jpg', 12000),
(45, 'Chavez', 'Table', 31, 1, 5, 0, 'default.jpg', 12000),
(46, 'Foreman', 'Hall', 90, 4, 163, 0, 'default.jpg', 12000),
(47, 'Kline', 'Hall', 81, 5, 260, 1, 'default.jpg', 12000),
(48, 'Good', 'Table', 14, 5, 2, 0, 'default.jpg', 12000),
(49, 'Kim', 'Hall', 24, 5, 203, 0, 'default.jpg', 12000),
(50, 'Burgess', 'Table', 38, 5, 10, 0, 'default.jpg', 12000),
(51, 'Gillespie', 'Hall', 98, 6, 285, 0, 'default.jpg', 12000),
(52, 'Evans', 'Hall', 38, 1, 317, 1, 'default.jpg', 12000),
(53, 'Gillespie', 'Hall', 79, 6, 425, 0, 'default.jpg', 12000),
(54, 'Mayo', 'Table', 7, 2, 5, 1, 'default.jpg', 12000),
(55, 'Klein', 'Hall', 12, 4, 91, 1, 'default.jpg', 12000),
(56, 'Mccarty', 'Table', 18, 6, 4, 1, 'default.jpg', 12000),
(57, 'Bradley', 'Hall', 92, 1, 250, 0, 'default.jpg', 12000),
(58, 'Larson', 'Hall', 62, 1, 355, 1, 'default.jpg', 12000),
(59, 'Kelly', 'Hall', 9, 1, 17, 0, 'default.jpg', 12000),
(60, 'Sutton', 'Table', 70, 2, 10, 0, 'default.jpg', 12000),
(61, 'Wise', 'Hall', 80, 2, 180, 1, 'default.jpg', 12000),
(62, 'Stevenson', 'Hall', 34, 2, 337, 1, 'default.jpg', 12000),
(63, 'Case', 'Table', 66, 1, 5, 0, 'default.jpg', 12000),

```
(64, 'Cash', 'Table', 88, 3, 4, 0, 'default.jpg', 12000),
(65, 'Day', 'Hall', 67, 4, 420, 0, 'default.jpg', 12000),
(66, 'Lopez', 'Table', 96, 5, 2, 1, 'default.jpg', 12000),
(67, 'Quinn', 'Hall', 44, 3, 99, 0, 'default.jpg', 12000),
(68, 'Hale', 'Hall', 2, 3, 140, 1, 'default.jpg', 12000),
(69, 'Townsend', 'Hall', 4, 2, 188, 1, 'default.jpg', 12000),
(70, 'Mclean', 'Table', 21, 2, 10, 1, 'default.jpg', 12000),
(71, 'Cunningham', 'Hall', 13, 6, 149, 1, 'default.jpg', 12000),
(72, 'Mccoy', 'Table', 16, 6, 5, 0, 'default.jpg', 12000),
(73, 'Landry', 'Hall', 32, 5, 189, 1, 'default.jpg', 12000),
(74, 'Keller', 'Hall', 13, 2, 274, 0, 'default.jpg', 12000),
(75, 'Cote', 'Hall', 11, 3, 39, 0, 'default.jpg', 12000),
(76, 'Osborn', 'Hall', 97, 1, 355, 0, 'default.jpg', 12000),
(77, 'Trujillo', 'Hall', 73, 3, 408, 0, 'default.jpg', 12000),
(78, 'Middleton', 'Table', 90, 4, 2, 0, 'default.jpg', 12000),
(79, 'Huber', 'Hall', 78, 5, 366, 1, 'default.jpg', 12000),
(80, 'Wynn', 'Table', 57, 6, 10, 1, 'default.jpg', 12000),
(81, 'Wood', 'Table', 83, 5, 5, 0, 'default.jpg', 12000),
(82, 'Monroe', 'Hall', 95, 6, 487, 0, 'default.jpg', 12000),
(83, 'Dudley', 'Hall', 59, 3, 412, 1, 'default.jpg', 12000);
```

Table structure for table `menu`

```
CREATE TABLE `menu` (
  `id` int(11) UNSIGNED NOT NULL,
  `item` varchar(255) NOT NULL,
  `price` int(5) NOT NULL,
  `hotel` int(11) UNSIGNED NOT NULL,
  `description` text,
  `img` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Dumping data for table `menu`

```
INSERT INTO `menu` (`id`, `item`, `price`, `hotel`, `description`, `img`)
VALUES
(1, 'Sushi', 500, 1, 'The best and special.', '1490069112-8778-f1.jpg'),
(2, 'Salad', 250, 1, 'The spicy one in our menu!', '1490069230-4007-f3.jpg'),
(3, 'Garlic Special', 100, 1, 'For those who love Gralic', '1490069271-8600-
f2.jpg'),
(4, 'Onion Special', 160, 1, 'Mouth watering.', '1490069300-3481-f4.jpg'),
(5, 'Cousine', 750, 1, 'Hotel Special', '1490069343-9633-f5.jpg');
```

Table structure for table `booking`

```
CREATE TABLE `booking` (
```

```
  `id` int(11) UNSIGNED NOT NULL,
  `account` int(11) UNSIGNED NOT NULL,
  `check_in` datetime NOT NULL,
  `check_out` datetime DEFAULT NULL,
  `facility` int(11) UNSIGNED NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Dumping data for table `booking`

```
INSERT INTO `booking` (`id`, `account`, `check_in`, `check_out`, `facility`)
VALUES
(1, 1, '2016-03-04 10:00:00', '2016-03-06 10:00:00', 1),
(2, 2, '2016-03-01 01:00:00', '2016-03-03 01:00:00', 2),
(3, 3, '2016-03-13 06:52:00', '2016-03-15 06:52:00', 3),
(4, 3, '2016-05-13 04:27:55', '2016-05-15 04:27:55', 11),
(5, 1, '2015-04-18 16:17:14', '2015-04-20 16:17:14', 11),
(6, 1, '2015-05-23 14:59:25', '2015-05-25 14:59:25', 20),
(7, 1, '2015-08-10 14:10:14', '2015-08-12 14:10:14', 14),
(8, 1, '2016-10-04 23:34:03', '2017-04-22 01:44:30', 13),
(9, 4, '2017-02-17 01:06:56', '2017-02-19 01:06:56', 11),
(10, 4, '2016-07-29 02:57:27', '2016-07-31 02:57:27', 20),
(11, 3, '2016-12-23 00:27:58', '2016-12-25 00:27:58', 12),
(12, 3, '2015-07-24 11:21:37', '2015-07-26 11:21:37', 20),
(13, 2, '2016-09-03 21:13:32', '2016-09-05 21:13:32', 9),
(14, 1, '2015-07-03 10:45:33', '2015-07-05 10:45:33', 16),
(15, 2, '2016-02-03 16:55:38', '2016-02-05 16:55:38', 13),
(16, 3, '2015-12-10 00:08:27', '2015-12-12 00:08:27', 13),
(17, 3, '2016-02-08 05:52:14', '2016-02-10 05:52:14', 5),
(18, 4, '2016-05-01 13:27:25', '2016-05-03 13:27:25', 3),
(19, 3, '2016-04-11 10:04:03', '2016-04-13 10:04:03', 19),
(20, 1, '2015-05-20 09:19:59', '2015-05-22 09:19:59', 20),
(21, 3, '2016-07-05 11:30:09', '2016-07-07 11:30:09', 14),
(22, 3, '2016-06-18 19:34:01', '2016-06-20 19:34:01', 20),
(23, 3, '2016-03-05 10:35:06', '2016-03-07 10:35:06', 5),
(24, 1, '2015-09-04 08:38:24', '2015-09-06 08:38:24', 13),
(25, 4, '2015-09-28 06:51:14', '2015-09-30 06:51:14', 4),
(26, 2, '2016-10-31 00:15:57', '2016-11-02 00:15:57', 20),
(27, 3, '2015-12-20 04:26:44', '2015-12-22 04:26:44', 15),
(28, 1, '2016-08-30 16:39:00', '2016-09-01 16:39:00', 8),
(29, 3, '2016-06-22 04:41:56', '2016-06-24 04:41:56', 1),
(30, 1, '2017-03-07 20:59:25', '2017-04-22 01:44:16', 9),
(31, 4, '2017-01-04 13:10:22', '2017-01-06 13:10:22', 14),
(32, 2, '2017-02-22 01:35:39', '2017-02-24 01:35:39', 11),
(33, 3, '2017-03-22 23:26:09', '2017-03-24 23:26:09', 18),
(34, 2, '2016-12-08 15:22:25', '2016-12-10 15:22:25', 3),
```

```
(35, 3, '2015-05-04 23:41:32', '2015-05-06 23:41:32', 16),
(36, 3, '2015-11-16 17:44:13', '2015-11-18 17:44:13', 4),
(37, 2, '2015-09-25 20:19:26', '2015-09-27 20:19:26', 2),
(38, 2, '2016-03-23 19:18:49', '2016-03-25 19:18:49', 17),
(39, 2, '2016-05-01 22:02:56', '2016-05-03 22:02:56', 7),
(40, 3, '2015-12-04 23:17:13', '2015-12-06 23:17:13', 8),
(41, 2, '2016-02-02 01:27:57', '2016-02-04 01:27:57', 2),
(42, 2, '2016-12-31 15:19:48', '2017-01-02 15:19:48', 3),
(43, 1, '2016-10-04 06:38:07', '2017-04-22 01:44:42', 9),
(44, 4, '2016-12-01 19:53:23', '2016-12-03 19:53:23', 8),
(45, 3, '2016-06-11 23:22:59', '2016-06-13 23:22:59', 8),
(46, 3, '2016-01-23 09:59:54', '2016-01-25 09:59:54', 13),
(47, 4, '2015-12-19 16:08:14', '2015-12-21 16:08:14', 4),
(48, 1, '2015-05-23 05:43:01', '2015-05-25 05:43:01', 4),
(49, 1, '2015-12-21 00:00:26', '2015-12-23 00:00:26', 7),
(50, 1, '2016-11-07 18:50:07', '2016-11-09 18:50:07', 2),
(51, 2, '2016-04-04 17:43:21', '2016-04-06 17:43:21', 2),
(52, 1, '2016-09-05 01:51:01', '2016-09-07 01:51:01', 17),
(53, 3, '2015-10-08 20:48:10', '2015-10-10 20:48:10', 7),
(54, 3, '2016-03-16 00:00:00', '2016-03-23 00:00:00', 48),
(55, 3, '2017-03-25 00:00:00', '2017-03-26 00:00:00', 4),
(56, 3, '2017-03-26 00:00:00', '2017-03-28 00:00:00', 48),
(57, 1, '2017-04-01 00:00:00', '2017-04-22 01:43:59', 42);
```

Table structure for table `order_booking`

```
CREATE TABLE `order_booking` (
  `id` int(11) UNSIGNED NOT NULL,
  `account` int(11) UNSIGNED NOT NULL,
  `booking` int(11) UNSIGNED NOT NULL,
  `date` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Dumping data for table `order_booking`

```
INSERT INTO `order_booking` (`id`, `account`, `booking`, `date`) VALUES
(1, 1, 43, '2016-06-12 08:47:07'),
(2, 1, 48, '2016-06-09 22:11:48'),
(3, 1, 30, '2016-06-13 11:42:11'),
(4, 1, 49, '2017-03-24 01:09:49'),
(5, 1, 50, '2016-06-28 22:42:23'),
(6, 1, 20, '2017-09-10 23:49:49'),
(7, 1, 1, '2017-04-20 18:22:25'),
(8, 1, 52, '2017-02-02 00:53:22'),
(9, 1, 14, '2016-06-20 21:56:21'),
(10, 1, 14, '2016-10-06 00:40:02'),
```

(11, 1, 30, '2016-09-03 03:45:26'),
(12, 1, 49, '2016-04-01 07:07:22'),
(13, 1, 20, '2017-08-08 23:06:03'),
(14, 1, 24, '2017-01-03 03:40:15'),
(15, 1, 1, '2017-03-11 10:13:52'),
(16, 1, 50, '2016-08-04 07:09:57'),
(17, 1, 52, '2017-05-11 18:07:04'),
(18, 1, 28, '2017-11-21 02:05:16'),
(19, 1, 24, '2018-02-19 18:01:10'),
(20, 1, 30, '2018-01-03 01:40:02'),
(21, 1, 5, '2016-07-21 17:39:05'),
(22, 1, 5, '2016-12-20 13:04:21'),
(23, 1, 6, '2016-03-30 23:23:00'),
(24, 1, 30, '2016-07-15 15:10:41'),
(25, 1, 6, '2017-01-09 04:55:46'),
(26, 1, 5, '2017-02-02 16:28:56'),
(27, 1, 28, '2017-12-15 01:58:10'),
(28, 1, 30, '2017-12-05 17:22:47'),
(29, 1, 1, '2016-06-03 23:56:47'),
(30, 1, 50, '2018-01-14 17:29:52'),
(31, 1, 28, '2017-05-31 07:00:46'),
(32, 1, 7, '2016-07-17 19:22:41'),
(33, 1, 49, '2017-07-24 18:26:51'),
(34, 1, 50, '2017-11-23 10:26:10'),
(35, 1, 20, '2017-05-30 18:35:33'),
(36, 1, 6, '2017-11-21 07:13:37'),
(37, 1, 14, '2017-08-04 08:36:46'),
(38, 1, 30, '2017-02-09 17:20:45'),
(39, 1, 49, '2017-04-11 13:43:13'),
(40, 1, 30, '2016-08-06 14:20:42'),
(41, 1, 43, '2018-03-06 18:15:51'),
(42, 1, 28, '2016-06-19 18:55:04'),
(43, 1, 8, '2018-03-10 22:26:24'),
(44, 1, 52, '2017-09-13 07:48:12'),
(45, 1, 5, '2016-10-31 03:31:19'),
(46, 1, 49, '2018-02-01 21:20:03'),
(47, 1, 50, '2016-05-05 21:46:41'),
(48, 1, 14, '2017-02-21 06:18:37'),
(49, 1, 50, '2018-01-28 10:05:42'),
(50, 1, 6, '2016-06-11 06:04:20'),
(51, 1, 43, '2017-03-24 08:44:52'),
(52, 1, 30, '2017-03-24 08:52:55'),
(53, 1, 30, '2017-03-24 08:53:28'),
(54, 1, 53, '2017-03-24 08:55:27'),
(55, 1, 53, '2017-03-24 09:00:07'),

```
(56, 1, 30, '2017-03-24 09:02:56'),
(57, 1, 30, '2017-03-24 09:05:41'),
(58, 1, 30, '2017-03-24 09:13:43'),
(59, 1, 30, '2017-03-25 17:17:22'),
(60, 1, 30, '2017-03-25 17:17:52'),
(61, 3, 55, '2017-03-25 17:45:48'),
(62, 3, 56, '2017-03-25 17:50:51'),
(63, 1, 30, '2017-04-02 13:16:18');
```

Table structure for table `order_contents`

```
CREATE TABLE `order_contents` (
  `id` int(11) UNSIGNED NOT NULL,
  `order_id` int(11) UNSIGNED NOT NULL,
  `menu_item` int(11) UNSIGNED NOT NULL,
  `quantity` int(5) NOT NULL,
  `status` enum('Booked','Under Process','On the way','Delivered','Failed') NOT
NULL DEFAULT 'Booked'
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Dumping data for table `order_contents`

```
INSERT INTO `order_contents` (`id`, `order_id`, `menu_item`, `quantity`,
`status`) VALUES
(1, 1, 1, 1, 'Failed'),
(2, 20, 1, 10, 'Failed'),
(3, 37, 2, 9, 'Delivered'),
(4, 13, 5, 6, 'Delivered'),
(5, 49, 3, 4, 'Under Process'),
(6, 28, 1, 8, 'Failed'),
(7, 42, 5, 1, 'On the way'),
(8, 32, 3, 3, 'Delivered'),
(9, 43, 3, 9, 'Failed'),
(10, 50, 3, 1, 'Delivered'),
(11, 2, 2, 1, 'On the way'),
(12, 40, 3, 6, 'Failed'),
(13, 42, 3, 7, 'Delivered'),
(14, 7, 4, 1, 'On the way'),
(15, 6, 4, 8, 'Under Process'),
(16, 39, 2, 4, 'Delivered'),
(17, 14, 3, 7, 'Under Process'),
(18, 1, 1, 10, 'Delivered'),
(19, 12, 3, 4, 'Delivered'),
(20, 27, 5, 5, 'On the way'),
(21, 19, 1, 10, 'Delivered'),
(22, 48, 3, 5, 'On the way'),
```

(23, 35, 2, 5, 'On the way'),
(24, 23, 4, 1, 'Booked'),
(25, 31, 1, 3, 'On the way'),
(26, 17, 2, 8, 'Booked'),
(27, 31, 2, 9, 'Booked'),
(28, 18, 3, 10, 'Under Process'),
(29, 20, 3, 2, 'Delivered'),
(30, 27, 3, 2, 'On the way'),
(31, 40, 5, 3, 'Delivered'),
(32, 15, 2, 6, 'Delivered'),
(33, 12, 4, 10, 'Delivered'),
(34, 22, 4, 1, 'On the way'),
(35, 15, 4, 1, 'Under Process'),
(36, 48, 2, 1, 'Delivered'),
(37, 48, 4, 1, 'On the way'),
(38, 23, 5, 2, 'Under Process'),
(39, 40, 4, 9, 'Failed'),
(40, 2, 3, 9, 'On the way'),
(41, 12, 2, 10, 'Under Process'),
(42, 38, 4, 8, 'Delivered'),
(43, 9, 2, 10, 'Delivered'),
(44, 30, 1, 9, 'Under Process'),
(45, 30, 5, 10, 'On the way'),
(46, 12, 2, 3, 'On the way'),
(47, 38, 5, 3, 'Delivered'),
(48, 13, 4, 5, 'Booked'),
(49, 30, 5, 8, 'Delivered'),
(50, 44, 5, 2, 'Delivered'),
(51, 27, 3, 3, 'Booked'),
(52, 51, 1, 52, 'Failed'),
(53, 51, 1, 4, 'Failed'),
(54, 52, 1, 1, 'Failed'),
(55, 52, 1, 3, 'Failed'),
(56, 53, 1, 1, 'Failed'),
(57, 54, 1, 1, 'Booked'),
(58, 55, 1, 5, 'Delivered'),
(59, 55, 1, 3, 'Booked'),
(60, 56, 1, 10, 'Delivered'),
(61, 57, 1, 1, 'Failed'),
(62, 58, 3, 5, 'Failed'),
(63, 58, 5, 1, 'Failed'),
(64, 59, 4, 3, 'Failed'),
(65, 60, 2, 1, 'Delivered'),
(66, 61, 4, 7, 'Booked'),
(67, 62, 1, 2, 'Booked'),

```
(68, 63, 1, 1, 'Failed'),
(69, 63, 2, 1, 'Failed');
```

Table structure for table `discount_coupons`

```
CREATE TABLE `discount_coupons` (
  `id` int(11) UNSIGNED NOT NULL,
  `coupon` varchar(255) DEFAULT 'You get',
  `valid_through` date NOT NULL,
  `valid_for` int(11) UNSIGNED NOT NULL,
  `type` varchar(255) NOT NULL,
  `value` varchar(255) NOT NULL,
  `status` enum('Available','Redeemed','Expired') NOT NULL,
  `img` varchar(255) DEFAULT 'default.png'
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Dumping data for table `discount_coupons`

```
INSERT INTO `discount_coupons` (`id`, `coupon`, `valid_through`,
`valid_for`, `type`, `value`, `status`, `img`) VALUES
(1, 'You get', '2017-03-10', 0, 'special', '50', 'Available', 'default.png'),
(2, 'You get', '2017-03-10', 2, 'rate cutter', '5', 'Available', 'default.png'),
(3, 'You get', '2018-02-05', 3, 'Special', '2', 'Expired', 'default.png'),
(4, 'You get', '2017-04-13', 4, 'Flat', '2', 'Available', 'default.png'),
(5, 'You get', '2017-07-29', 4, 'Flat', '5', 'Redeemed', 'default.png'),
(6, 'You get', '2018-01-16', 3, 'Flat', '25', 'Redeemed', 'default.png'),
(7, 'You get', '2017-04-07', 3, 'Rate Cutter', '25', 'Expired', 'default.png'),
(8, 'You get', '2017-09-06', 1, 'Flat', '20', 'Redeemed', 'default.png'),
(9, 'You get', '2017-08-12', 1, 'Special', '20', 'Expired', 'default.png'),
(10, 'You get', '2017-04-01', 4, 'Special', '2', 'Redeemed', 'default.png'),
(11, 'You get', '2017-05-24', 0, 'Flat', '25', 'Available', 'default.png'),
(12, 'You get', '2018-01-30', 2, 'Flat', '5', 'Available', 'default.png'),
(13, 'You get', '2017-04-21', 1, 'Flat', '2', 'Redeemed', 'default.png'),
(14, 'You get', '2017-03-06', 4, 'Flat', '25', 'Expired', 'default.png'),
(15, 'You get', '2018-01-11', 3, 'Special', '25', 'Redeemed', 'default.png'),
(16, 'You get', '2017-10-31', 4, 'Rate Cutter', '2', 'Available', 'default.png'),
(17, 'You get', '2017-11-01', 1, 'Rate Cutter', '20', 'Available', 'default.png'),
(18, 'You get', '2017-12-24', 2, 'Special', '25', 'Available', 'default.png'),
(19, 'You get', '2017-06-06', 3, 'Special', '25', 'Available', 'default.png'),
(20, 'You get', '2018-03-15', 3, 'Rate Cutter', '10', 'Redeemed', 'default.png'),
(21, 'You get', '2017-12-15', 3, 'Flat', '25', 'Redeemed', 'default.png'),
(22, 'You get', '2018-02-14', 2, 'Rate Cutter', '25', 'Expired', 'default.png'),
(23, 'You get', '2017-08-18', 2, 'Flat', '10', 'Available', 'default.png'),
(24, 'You get', '2017-07-30', 0, 'Special', '10', 'Available', 'default.png'),
(25, 'You get', '2017-11-24', 4, 'Rate Cutter', '5', 'Expired', 'default.png'),
(26, 'You get', '2017-10-27', 0, 'Rate Cutter', '10', 'Expired', 'default.png'),
```

(27, 'You get', '2018-02-12', 1, 'Special', '5', 'Expired', 'default.png'),
(28, 'You get', '2017-03-04', 1, 'Special', '10', 'Available', 'default.png'),
(29, 'You get', '2018-01-19', 1, 'Flat', '25', 'Expired', 'default.png'),
(30, 'You get', '2018-02-23', 1, 'Special', '5', 'Available', 'default.png'),
(31, 'You get', '2017-08-15', 2, 'Special', '5', 'Available', 'default.png'),
(32, 'You get', '2017-05-20', 0, 'Rate Cutter', '2', 'Redeemed', 'default.png'),
(33, 'You get', '2017-11-05', 4, 'Special', '20', 'Expired', 'default.png'),
(34, 'You get', '2018-02-19', 3, 'Special', '5', 'Available', 'default.png'),
(35, 'You get', '2018-02-15', 1, 'Rate Cutter', '25', 'Redeemed', 'default.png'),
(36, 'You get', '2017-09-03', 2, 'Rate Cutter', '25', 'Available', 'default.png'),
(37, 'You get', '2017-07-25', 0, 'Special', '25', 'Available', 'default.png'),
(38, 'You get', '2017-12-26', 1, 'Flat', '2', 'Available', 'default.png'),
(39, 'You get', '2017-03-25', 2, 'Rate Cutter', '20', 'Available', 'default.png'),
(40, 'You get', '2017-03-13', 1, 'Flat', '25', 'Available', 'default.png'),
(41, 'You get', '2017-08-06', 3, 'Rate Cutter', '2', 'Redeemed', 'default.png'),
(42, 'You get', '2017-07-27', 0, 'Special', '5', 'Redeemed', 'default.png'),
(43, 'You get', '2017-03-04', 1, 'Flat', '10', 'Redeemed', 'default.png'),
(44, 'You get', '2017-09-09', 4, 'Special', '10', 'Redeemed', 'default.png'),
(45, 'You get', '2017-06-14', 3, 'Rate Cutter', '20', 'Available', 'default.png'),
(46, 'You get', '2017-06-06', 1, 'Special', '2', 'Expired', 'default.png'),
(47, 'You get', '2017-05-10', 1, 'Rate Cutter', '2', 'Available', 'default.png'),
(48, 'You get', '2017-05-14', 1, 'Rate Cutter', '10', 'Expired', 'default.png'),
(49, 'You get', '2017-04-30', 0, 'Flat', '5', 'Redeemed', 'default.png'),
(50, 'You get', '2017-06-18', 0, 'Rate Cutter', '20', 'Expired', 'default.png'),
(51, 'You get', '2017-12-15', 0, 'Flat', '2', 'Available', 'default.png'),
(52, 'You get', '2017-12-07', 0, 'Special', '25', 'Expired', 'default.png'),
(53, 'You get', '2017-11-14', 4, 'Special', '25', 'Available', 'default.png'),
(54, 'You get', '2017-08-23', 2, 'Special', '5', 'Expired', 'default.png'),
(55, 'You get', '2017-06-27', 3, 'Rate Cutter', '20', 'Available', 'default.png'),
(56, 'You get', '2018-02-05', 3, 'Special', '5', 'Available', 'default.png'),
(57, 'You get', '2017-03-12', 4, 'Special', '20', 'Available', 'default.png'),
(58, 'You get', '2017-08-08', 2, 'Flat', '5', 'Available', 'default.png'),
(59, 'You get', '2017-07-30', 0, 'Rate Cutter', '25', 'Expired', 'default.png'),
(60, 'You get', '2017-09-13', 0, 'Rate Cutter', '10', 'Redeemed', 'default.png'),
(61, 'You get', '2017-03-15', 4, 'Rate Cutter', '2', 'Expired', 'default.png'),
(62, 'You get', '2017-05-23', 3, 'Special', '5', 'Available', 'default.png'),
(63, 'You get', '2018-02-08', 0, 'Rate Cutter', '10', 'Available', 'default.png'),
(64, 'You get', '2017-03-10', 3, 'Rate Cutter', '25', 'Expired', 'default.png'),
(65, 'You get', '2017-09-21', 4, 'Rate Cutter', '25', 'Redeemed', 'default.png'),
(66, 'You get', '2017-08-13', 1, 'Rate Cutter', '25', 'Expired', 'default.png'),
(67, 'You get', '2017-03-27', 4, 'Rate Cutter', '20', 'Expired', 'default.png'),
(68, 'You get', '2017-04-07', 2, 'Rate Cutter', '10', 'Expired', 'default.png'),
(69, 'You get', '2017-05-24', 4, 'Flat', '25', 'Available', 'default.png'),
(70, 'You get', '2017-09-28', 4, 'Special', '25', 'Redeemed', 'default.png'),
(71, 'You get', '2017-08-23', 3, 'Rate Cutter', '2', 'Available', 'default.png'),

```
(72, 'You get', '2017-11-18', 3, 'Rate Cutter', '10', 'Available', 'default.png'),
(73, 'You get', '2017-03-28', 4, 'Special', '2', 'Redeemed', 'default.png'),
(74, 'You get', '2017-09-03', 3, 'Flat', '25', 'Available', 'default.png'),
(75, 'You get', '2017-06-15', 1, 'Special', '2', 'Available', 'default.png'),
(76, 'You get', '2017-06-27', 4, 'Flat', '2', 'Expired', 'default.png'),
(77, 'You get', '2018-02-02', 4, 'Flat', '10', 'Available', 'default.png'),
(78, 'You get', '2018-01-14', 4, 'Flat', '20', 'Expired', 'default.png'),
(79, 'You get', '2017-07-28', 2, 'Special', '5', 'Available', 'default.png'),
(80, 'You get', '2017-07-03', 1, 'Rate Cutter', '10', 'Expired', 'default.png'),
(81, 'You get', '2018-02-16', 4, 'Rate Cutter', '25', 'Expired', 'default.png'),
(82, 'You get', '2017-08-06', 1, 'Special', '25', 'Redeemed', 'default.png'),
(83, 'You get', '2017-11-04', 3, 'Rate Cutter', '5', 'Expired', 'default.png'),
(84, 'You get', '2017-06-06', 0, 'Flat', '20', 'Redeemed', 'default.png'),
(85, 'You get', '2017-06-25', 3, 'Rate Cutter', '10', 'Expired', 'default.png'),
(86, 'You get', '2017-07-24', 0, 'Flat', '10', 'Redeemed', 'default.png'),
(87, 'You get', '2017-11-09', 0, 'Rate Cutter', '2', 'Redeemed', 'default.png'),
(88, 'You get', '2017-05-06', 2, 'Flat', '5', 'Redeemed', 'default.png'),
(89, 'You get', '2017-03-20', 3, 'Special', '10', 'Expired', 'default.png'),
(90, 'You get', '2017-11-06', 0, 'Special', '20', 'Available', 'default.png'),
(91, 'You get', '2018-01-15', 3, 'Rate Cutter', '10', 'Available', 'default.png'),
(92, 'You get', '2017-06-16', 3, 'Special', '25', 'Expired', 'default.png'),
(93, 'You get', '2018-01-27', 2, 'Flat', '10', 'Expired', 'default.png'),
(94, 'You get', '2017-04-21', 4, 'Flat', '25', 'Available', 'default.png'),
(95, 'You get', '2017-09-21', 3, 'Flat', '20', 'Redeemed', 'default.png'),
(96, 'You get', '2017-05-26', 4, 'Rate Cutter', '10', 'Expired', 'default.png'),
(97, 'You get', '2017-07-27', 4, 'Flat', '5', 'Available', 'default.png'),
(98, 'You get', '2018-01-11', 2, 'Flat', '20', 'Expired', 'default.png'),
(99, 'You get', '2017-11-21', 4, 'Rate Cutter', '25', 'Expired', 'default.png'),
(100, 'You get', '2017-07-20', 4, 'Special', '5', 'Redeemed', 'default.png'),
(101, 'You get', '2018-01-15', 0, 'Special', '2', 'Expired', 'default.png'),
(102, 'You get', '2017-03-03', 3, 'Rate Cutter', '2', 'Available', 'default.png');
```

Table structure for table `invoice`

```
CREATE TABLE `invoice` (
  `id` int(100) UNSIGNED NOT NULL,
  `booking` int(11) UNSIGNED NOT NULL,
  `generation_date` datetime NOT NULL,
  `total_amount` int(5) NOT NULL,
  `discount` int(11) NOT NULL,
  `amount_payable` int(11) NOT NULL,
  `status` enum('Generated','Paid','Over due','Waived') NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Dumping data for table `invoice`

```
INSERT INTO `invoice` (`id`, `booking`, `generation_date`, `total_amount`,
`discount`, `amount_payable`, `status`) VALUES
(1, 50, '2017-03-09 01:05:00', 5000, 1, 2500, 'Paid'),
(2, 57, '2017-04-22 01:43:59', 252000, 38, 251996, 'Paid'),
(3, 30, '2017-04-22 01:44:16', 551230, 30, 493017, 'Paid'),
(4, 8, '2017-04-22 01:44:30', 2388000, 75, 2293435, 'Paid'),
(5, 43, '2017-04-22 01:44:42', 2405000, 17, 2399960, 'Paid');
```

Indexes for dumped tables

Indexes for table `account`

```
ALTER TABLE `account`
  ADD PRIMARY KEY (`id`);
```

Indexes for table `booking`

```
ALTER TABLE `booking`
  ADD PRIMARY KEY (`id`);
```

Indexes for table `discount_coupons`

```
ALTER TABLE `discount_coupons`
  ADD PRIMARY KEY (`id`),
  ADD KEY `couponfor_idx` (`valid_for`);
```

Indexes for table `facility`

```
ALTER TABLE `facility`
  ADD PRIMARY KEY (`id`);
```

Indexes for table `hotel`

```
ALTER TABLE `hotel`
  ADD PRIMARY KEY (`id`);
```

Indexes for table `invoice`

```
ALTER TABLE `invoice`
  ADD PRIMARY KEY (`id`),
  ADD KEY `generated_idx` (`booking`);
```

Indexes for table `menu`

```
ALTER TABLE `menu`
 ADD PRIMARY KEY (`id`);
```

Indexes for table `order_booking`

```
ALTER TABLE `order_booking`
 ADD PRIMARY KEY (`id`),
 ADD KEY `user_idx` (`account`),
 ADD KEY `forbooking_idx` (`booking`);
```

Indexes for table `order_contents`

```
ALTER TABLE `order_contents`
 ADD PRIMARY KEY (`id`),
 ADD KEY `contents_of_idx` (`order_id`),
 ADD KEY `content_item_idx` (`menu_item`);
```

Indexes for table `person`

```
ALTER TABLE `person`
 ADD PRIMARY KEY (`id`);
```

Constraints for dumped tables

Constraints for table `invoice`

```
ALTER TABLE `invoice`
 ADD CONSTRAINT `generated` FOREIGN KEY (`booking`) REFERENCES
`booking` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION;
```

Constraints for table `order_booking`

```
ALTER TABLE `order_booking`
 ADD CONSTRAINT `forbooking` FOREIGN KEY (`booking`) REFERENCES
`booking` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
 ADD CONSTRAINT `user` FOREIGN KEY (`account`) REFERENCES `account`
(`id`) ON DELETE NO ACTION ON UPDATE NO ACTION;
```

Constraints for table `order_contents`

```
ALTER TABLE `order_contents`
 ADD CONSTRAINT `item` FOREIGN KEY (`menu_item`) REFERENCES `menu`
(`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
```

ADD CONSTRAINT `ordered` FOREIGN KEY (`order_id`) REFERENCES `order_booking` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION;

Database Data

Account

```
SELECT * FROM `account`
```

id	user_name	password	authority_level	person
1	mandeep	shabina	3	1
2	lakshay	8946553	3	2
3	mandyk	123456	1	1
4	vipul1	123456	1	3
5	everyone	supass	1	5

Booking

```
SELECT * FROM `booking`
```

id	account	check_in	check_out	facility
1	1	2016-03-04 10:00:00	2016-03-06 10:00:00	1
2	2	2016-03-01 01:00:00	2016-03-03 01:00:00	2
3	3	2016-03-13 06:52:00	2016-03-15 06:52:00	3
4	3	2016-05-13 04:27:55	2016-05-15 04:27:55	11
5	1	2015-04-18 16:17:14	2015-04-20 16:17:14	11
6	1	2015-05-23 14:59:25	2015-05-25 14:59:25	20
7	1	2015-08-10 14:10:14	2015-08-12 14:10:14	14
8	1	2017-04-15 00:00:03	2017-04-24 03:43:55	13
9	4	2017-02-17 01:06:56	2017-02-19 01:06:56	11
10	4	2016-07-29 02:57:27	2016-07-31 02:57:27	20
11	3	2016-12-23 00:27:58	2016-12-25 00:27:58	12
12	3	2015-07-24 11:21:37	2015-07-26 11:21:37	20
13	2	2016-09-03 21:13:32	2016-09-05 21:13:32	9
14	1	2015-07-03 10:45:33	2015-07-05 10:45:33	16
15	2	2016-02-03 16:55:38	2016-02-05 16:55:38	13
16	3	2015-12-10 00:08:27	2015-12-12 00:08:27	13
17	3	2016-02-08 05:52:14	2016-02-10 05:52:14	5
18	4	2016-05-01 13:27:25	2016-05-03 13:27:25	3
19	3	2016-04-11 10:04:03	2016-04-13 10:04:03	19
20	1	2015-05-20 09:19:59	2015-05-22 09:19:59	20
21	3	2016-07-05 11:30:09	2016-07-07 11:30:09	14
22	3	2016-06-18 19:34:01	2016-06-20 19:34:01	20
23	3	2016-03-05 10:35:06	2016-03-07 10:35:06	5
24	1	2015-09-04 08:38:24	2015-09-06 08:38:24	13
25	4	2015-09-28 06:51:14	2015-09-30 06:51:14	4

Discount Coupons

```
SELECT * FROM `discount_coupons`
```

id	coupon	valid_through	valid_for	type	value	status	img
1	You get	2017-03-10	0	special	50	Available	default.png
2	Discount	2017-03-10	2	rate cutter	5	Available	default.png
3	You get	2018-02-05	3	Special	2	Expired	default.png
4	WOw	2017-04-13	4	Flat	2	Available	default.png
5	You get	2017-07-29	4	Flat	5	Redeemed	default.png
6	Awesome	2018-01-16	3	Flat	25	Redeemed	default.png
7	Great	2017-04-07	3	Rate Cutter	25	Expired	default.png
8	You get	2017-09-06	1	Flat	20	Redeemed	default.png
9	Superb	2017-08-12	1	Special	20	Expired	default.png
10	You get	2017-04-01	4	Special	2	Redeemed	default.png
11	Yummy deals	2017-05-24	0	Flat	25	Available	default.png
12	You get	2018-01-30	2	Flat	5	Available	default.png

Facility

```
SELECT * FROM `facility`
```

id	title	type	number	floor	capacity	available	img	charges
1	Deluxe	Hall	2	5	50	1	1490062697-5629-c1.jpg	12000
2	Beach Side	Golden Room	1	1	5	1	1490062720-8617-c2.jpg	12000
3	Pool side	Silver Room	1	1	2	1	1490062753-6854-c3.jpg	12000
4	Cummings	Diamond Room	30	2	2	0	default.jpg	12000
5	Pitts	Hall	63	4	50	0	default.jpg	12000
6	Santiago	Golden Room	59	6	2	0	default.jpg	12000
7	Valenzuela	Table	25	1	5	0	default.jpg	12000
8	Conner	Diamond Room	58	6	2	1	default.jpg	12000
9	Irwin	Golden Room	57	1	2	1	default.jpg	12000
10	Wright	Table	25	4	10	0	default.jpg	12000
11	Garrison	Silver Room	21	5	2	1	default.jpg	12000
12	Miranda	Golden Room	40	4	2	1	default.jpg	12000
13	Porter	Silver Room	48	1	2	1	default.jpg	12000
14	Mclaughlin	Table	67	5	5	1	default.jpg	12000
15	Palmer	Golden Room	88	4	2	0	default.jpg	12000

Invoice

```
SELECT * FROM `invoice`
```

id	booking	generation_date	total_amount	discount	amount_payable	status
1	50	2017-03-09 01:05:00	5000	1	2500	Paid
2	57	2017-04-22 01:43:59	252000	38	251996	Paid
3	30	2017-04-22 01:44:16	551230	30	493017	Paid
5	43	2017-04-22 01:44:42	2405000	17	2399960	Paid
6	8	2017-04-24 03:43:55	108000	30	97470	Paid
7	58	2017-04-24 03:53:13	3480	24	2818	Paid

Order Booking

```
SELECT * FROM `order_booking`
```

id	account	booking	date
1	1	43	2016-06-12 08:47:07
2	1	48	2016-06-09 22:11:48
3	1	30	2016-06-13 11:42:11
4	1	49	2017-03-24 01:09:49
5	1	50	2016-06-28 22:42:23
6	1	20	2017-09-10 23:49:49
7	1	1	2017-04-20 18:22:25
8	1	52	2017-02-02 00:53:22
9	1	14	2016-06-20 21:56:21
10	1	14	2016-10-06 00:40:02
11	1	30	2016-09-03 03:45:26
12	1	49	2016-04-01 07:07:22
13	1	20	2017-08-08 23:06:03

Order Contents

```
SELECT * FROM `order_contents`
```

id	order_id	menu_item	quantity	status
1	1	1	1	Failed
2	20	1	10	Failed
3	37	2	9	Delivered
4	13	5	6	Delivered
5	49	3	4	Under Process
6	28	1	8	Failed
7	42	5	1	On the way
8	32	3	3	Delivered
9	43	3	9	Failed
10	50	3	1	Delivered
11	2	2	1	On the way
12	40	3	6	Failed
13	42	3	7	Delivered

Person

```
SELECT * FROM `person`
```

id	first_name	last_name	address	phone	email	dob	anniversary	img
1	Mandeep	Kaur	65 Jalandhar	00002	mandeepshabina@gmail.com	1993-07-27	1993-07-27	1490066978-2223-p1.jpg
2	Lakshay	Verma	8, 14 Jalandhar Cantt	9779333346	verma_lakshay@live.in	1993-10-31	2011-03-25	1493006094-5059-clown.png
3	Vipul	Gupta	Adarsh Nagar Jalandhar	123485	vipulgupta@gmail.com	1993-10-15	2017-03-05	1490142904-9391-g1.jpg
4	Tarun	Veer	BASANT AVENUE	5555	tarun@gmail.com	1993-07-07	2017-03-03	1490142885-1297-images.png
5	Rohit	Kural	deep nagar	21	rohit@gmail.com	1990-04-29	2015-01-19	1490142930-6673-g3.jpg

Complete Project Coding

Client Pages

Index

```
<?php
$nav_only = FALSE;
include './layouts/header.php';
?>
<a id="mLocation" href="#hotel_location" class="scroll">
<!--    <span class="glyphicon glyphicon-home"></span>-->

   <img src="images/pin.png" width="32px" height="32px"/>

</a>

<div id="about" class="welcome">
   <div class="container">
      <div class="agile-title">
         <h3> Welcome !</h3>
      </div>
      <div class="w3ls-row">
         <div class="col-md-6 welcome-left">
            <div class="welcome-img">
               <img src="hotel_pics/lobby9.jpg" class="img-responsive zoom-
img" alt=""/>
            </div>
            <div class="col-md-6 welcome-left-grids">
               <div class="welcome-img">
                  <img src="hotel_pics/room1.jpg" class="img-responsive zoom-
img" alt=""/>
               </div>
            </div>
            <div class="col-md-6 welcome-left-grids">
               <div class="welcome-img">
                  <img src="hotel_pics/lobby3.jpg" class="img-responsive
zoom-img" alt=""/>
               </div>
            </div>
            <div class="clearfix"> </div>
         </div>
         <div class="col-md-6 welcome-right">
            <p>A good hotel shouldn't be necessarily expensive. Check this out
yourself. Book a room and get all possible comfort of a five star hotel. We work
for you to enjoy your stay.A hotel is an establishment that provides paid lodging
on a short-term basis. Facilities provided may range from a modest-quality
```

```
mattress in a small room to large suites with bigger, higher-quality beds, a
dresser, a fridge and other kitchen facilities, upholstered chairs, a flat screen
television and en-suite bathrooms.</p>
            <div class="open-hours-row">
               <div class="col-md-3 open-hours-left">
                  <h4>MANN HOTEL</h4>
               </div>
               <div class="col-md-3 open-hours-left">
                  <h6>ROOM</h6>
                  <h5>Starting at Rs 1,500</h5>
               </div>
               <div class="col-md-3 open-hours-left">
                  <h6>PARTY HALL</h6>
                  <h5>Starting at Rs 20,000</h5>
               </div>
               <div class="col-md-3 open-hours-left">
                  <h6>TABLE</h6>
                  <h5>Starting at Rs 500</h5>
               </div>
               <div class="clearfix"> </div>
            </div>
         </div>
         <div class="clearfix"> </div>
      </div>
   </div>
</div>
<!-- //welcome -->

<!-- slid -->
<div class="slid">
   <div class="container">
      <h4>Make Your Stay Cool</h4>
      <h5> Find <span>Offers</span></h5>
      <p>
         One of the most important things we want to achieve is being the most
comfortable, friendly and luxurious travel destination. And we keep it that way
for a long time!
      </p>
   </div>
</div>
<!-- //slid -->
<!-- menu -->
<div id="menu" class="menu">
   <div class="container">
      <div class="agile-title">
```

```
            <h3> Our Menu</h3>
        </div>
        <ul class="accordion">
            <?php
            $menu = Menu::find_expensive(5);
            $counter = 1;
            while ($item = current($menu)):
                ?>
                <li class="slide-<?php echo $counter; ?>">
                    <div class="menu-left">
                        <img src="<?php echo $item->img(); ?>" alt=""/>
                        <div class="menu-right">
                            <h4><?php echo $item->name(); ?> </h4>
                            <h5>₹ <?php echo $item->price * 10; ?> </h5>
                            <p><?php echo $item->description; ?></p>
                        </div>
                    </div>
                </li>
                <?php
                next($menu);
                $counter++;
            endwhile;
            ?>
        </ul>
        <div class="clearfix"> </div>
    </div>
</div>
<!-- //menu -->
<!-- team -->
<div id="team" class="team">
    <div class="container">
        <div class="agile-title">
            <h3> Our Team</h3>
        </div>
        <div class="team-row">
            <?php
            $objects = Account::find_admins(5);
            while ($object = current($objects)):
                ?>

                <div class="col-md-3 team-grids">
                    <div class="w3ls-effect">
                        <?php echo $object->avatar("72px"); ?>
                        <div class="view-caption">
                            <h4><?php echo $object->name(); ?></h4>
```

```
                    <p><?php echo $object->user_name; ?></p>
                </div>
            </div>
        </div>
        <?php
        next($objects);
    endwhile;
    ?>

    <div class="clearfix"> </div>
  </div>
 </div>
</div>
<!-- //team -->
<!-- facility's -->
<div id="facility" class="facility">
  <div class="container">
    <div class="agile-title">
      <h3>Facilities Available</h3>
    </div>
    <div class="gallery-agileinfo-row">

      <?php
      $facilities = Facility::show_case();
      while ($facility = current($facilities)):
        ?>
        <div class="col-md-4 gallery-grids">
          <div class="hover ehover14">
            <a href="" class="swipebox" title="">
              <img src="<?php echo $facility->img(); ?>"/>
              <div class="overlay">
                <h4><?php echo $facility->name(); ?></h4>
                <div class="info nullbutton button" data-toggle="modal"
data-target="#modal14">Show More</div>
              </div>
            </a>
          </div>
        </div>
        <?php
        next($facilities);
    endwhile;
    ?>
    <div class="clearfix"> </div>
  </div>
```

```
</div>
<!-- //facility's -->
<?php include './layouts/footer.php'; ?>
```

Login

```
<?php
$nav_only = TRUE;
include './layouts/header.php';

if ($session->is_logged_in()) {
   redirect_to("index.php");
}

if (isset($_POST['username']) && isset($_POST['password'])) {
   $user = Account::find_by_sql(
               "select * from account "
               . "where"
               . " user_name = \"{$_POST['username']}\""
               . " and"
               . " password = \"{$_POST['password']}\""
   );
   $user = array_shift($user);
   if (isset($user->id)) {
      $session->login($user);
      if (empty($_POST['request_uri'])) {
         redirect_to('index.php');
      } else {
         redirect_to($_POST['request_uri']);
      }
   } else {
      $message = "Incorrect credentials";
   }
}
?>

<style>
form {
   border: 3px solid #f1f1f1;
}

input[type=text], input[type=password] {
   width: 40%;
   padding: 10px 15px;
   margin: 8px 0;
```

```
    display: inline-block;
    border: 2px solid #ccc;
    box-sizing: border-box;
}

button {
    background-color: black;
    color: white;
    padding: 10px 15px;
    margin: 8px 0;
    border: none;
    cursor: pointer;
    width: 40%;
}

button:hover {
    opacity: 0.8;
}

.cancelbtn {
    width: auto;
    padding: 10px 18px;
    background-color: #f44336;
}

.imgcontainer {
    text-align: center;
    margin: 24px 0 12px 0;
}

img.avatar {
    width: 40%;
    border-radius: 50%;
}

.container {
    padding: 16px;
}

span.psw {
    float: right;
    padding-top: 16px;
}

/* Change styles for span and cancel button on extra small screens */
```

```
@media screen and (max-width: 300px) {
   span.psw {
      display: block;
      float: none;
   }
   .cancelbtn {
      width: 100%;
   }
}
</style>
<DIV class="">
   <br>
   <br>

<h2 align="center">Login Form</h2>

<form action="./login.php" method="post">
  <div class="imgcontainer">
     <img src="hotel_pics/hotels-4.jpg" alt="Avatar" class="avatar">
  </div>

  <div class="container">
     <div class="col-sm-8 col-sm-offset-4"><b>Username</b>
   <input type="text" placeholder="Enter Username" name="username"
required>
     </div>


   <div class="col-sm-8 col-sm-offset-4"><b>Password</b>
   <input type="password" placeholder="Enter Password" name="password"
required>
   </div>
     <div class="col-sm-10 col-sm-offset-4">
   <button type="submit" >Login</button>
     </div>
  </div>
</form>

</DIV>

<?php include './layouts/footer.php'; ?>
<script>
   var activeForm = '#login-form';

   $(function () {
```

```
        $('#login-form-link').click(function (e) {
            $("#login-form").delay(100).fadeIn(100);
            $("#register-form").fadeOut(100);
            $('#register-form-link').removeClass('active btn-primary');
            $(this).addClass('btn-primary');
            activeForm = "#register-form";
            e.preventDefault();
        });
        $('#register-form-link').click(function (e) {
            $("#register-form").delay(100).fadeIn(100);
            $("#login-form").fadeOut(100);
            $('#login-form-link').removeClass('active btn-primary');
            $(this).addClass('btn-primary');
            activeForm = "#login-form";
            e.preventDefault();
        });
    });

    $(activeForm).validate();
</script>
```

Logout

```
<?php
$nav_only = false;
include './layouts/header.php';
$session->logout();
?>

<script>
    window.location.href = 'index.php';
</script>
```

My Booking

```
<?php
$nav_only = TRUE;
include './layouts/header.php';
members_only();
$bookings = Booking::find_all_for_account($current_user->id);
$booking_id = (isset($_GET['booking'])) ? $_GET['booking'] : current($bookings)-
>id;
//$current_booking = Booking::find_by_id($booking_id);
//$current_booking = ($current_booking) ? $current_booking :
current($bookings);
?>
```

```
<div class="container-fluid">
   <!--Bookings Navigation-->
   <div class="col-md-3">
      <ul class="list-group bookings">
         <li class="list-group-item mandy">
            <a href="#" class="mandy-btn" data-toggle="modal" data-
target="#myModal">Book a New Facility</a>
         </li>

         <?php
         while ($booking = current($bookings)) {
            $booking->init_members();
            ?>
            <li id="booking-<?php echo $booking->id ?>" class="w3ls-row list-
group-item mandy" >
               <a class="selector_mk" onclick="get_details(<?php echo
$booking->id; ?>); return false;">
                  <span class="list-group-item-heading"><?php echo $booking-
>facilityObj->avatar("72px", "zoom-img"); ?></span>
                  <h4 style="display: inline-block;"><?php echo $booking-
>facilityObj->name(); ?></h4>
                  <br>
                  <span class="text-center text-info"><em>(<?php echo
$booking->check_in(); ?>)</em></span>
               </a>
            </li>
            <?php
            next($bookings);
         }
         ?>
      </ul>
      <?php include './layouts/booking_form.php'; ?>
   </div>
   <!--Booking Navigation End-->

   <!--Booking Orders-->
   <div id="booking_details" class="col-md-8"></div>
   <!--Booking Orders End-->
</div>
<?php include './layouts/footer.php'; ?>

<script>
   get_details(<?php echo $booking_id; ?>);

</script>
```

Checkout

```
<?php
$nav_only = TRUE;
$title = "Checkout";
include './layouts/header.php';
members_only();
$current_booking = Booking::find_by_id($session->current_booking());
if ($current_booking->id):
   $current_booking->init_members();
   $invoice = Invoice::make_for_booking($current_booking);
   ?>
   <section class="container-fluid">
      <article class="panel panel-default col-md-8 col-md-offset-2">
         <div id="booking_details" class="panel-body"></div>
      </article>
      <article id="checkout" class="panel panel-primary col-md-8 col-md-
offset-2">
         <header class="panel-heading">
            <h3>Checkout.</h3>
         </header>
         <div class="panel-body">
            <blockquote>
               All bills are system generated, you just need to press Pay and
Relax.
            </blockquote>
            <footer class="panel-footer">

               <table class="table table-hover">
                  <tr>
                     <td>
                        Estimated Bill
                     </td>
                     <td>
                        <?= SITE_CURRENCY . $current_booking->bill . "/-"; ?>
                     </td>

                  </tr>
                  <tr>
                     <td>
                        Billed Amount
                        <em>(Items that were delivered + Room Charges)</em>
                     </td>
                     <td>
```

```
                        <?= SITE_CURRENCY . $invoice->total_amount . "/-"; ?>
                    </td>
                </tr>
                <tr>
                    <td>
                        Apply a coupon
                    </td>

                    <td>
                        <select class="form-control" id="coupon" name="coupon"
onchange="calculate_bill()">
                            <option value="">Select a discount coupon</option>
                            <?php
                            $options =
Discount_coupons::find_appropriate_for($current_user);
                            include './layouts/options_list.php';
                            ?>
                        </select>

                        <script>
                            var bill = <?= $invoice->total_amount; ?>;
                            function calculate_bill() {
                                var cpn = $("#coupon").val();
                                if (cpn !== "") {
                                    var url = "./logic/coupon_validations.php";
                                    var data = {
                                        coupon: cpn,
                                        booking: <?= $current_booking->id; ?>,
                                        amount: bill
                                    };

                                    $.post(url, data, function (data) {
                                        if (!data.error) {
                                            var am = "₹ " + data.bill + "/-";

                                            $("#coupon_id").val(cpn);
                                            $("#amount").val(data.bill);

                                            $(".ammount_payable").html(am);
                                        } else {
                                            $(".ammount_payable").html(bill);
                                            alert(data.msg);
                                        }
                                    }, 'json');
```

```
                    }

                }
            </script>

        </td>
    </tr>
    <tr>
        <td>After Discounts</td>
        <td class="ammount_payable">
            <?= SITE_CURRENCY . $invoice->amount_payable . "/-"; ?>
        </td>
    </tr>

    <tr>
        <td>
            <a href="#booking_details" class="btn btn-info">
                <span class="glyphicon glyphicon-arrow-up"></span>
                <span>Recheck the expenses.</span>
            </a>
        </td>
        <td>
            <form action="./logic/checkout.php" method="post">
                <input id="amount" name="amount" type="text"/>
                <input id="coupon_id" name="coupon_id"
type="text"/>
                <input id="booking_id" name="booking_id" type="text"
value="<?=$current_booking->id;?>"/>
                <button type="submit" class="btn btn-primary">
                    <span class="glyphicon glyphicon-check"></span>
                    Pay <span class="ammount_payable"><?=
SITE_CURRENCY . $invoice->amount() . "/-"; ?></span>
                </button>
            </form>
        </td>
    </tr>
</table>
</footer>
</div>
</article>
</section>
<?php endif; ?>
<script>
    get_details(<?php echo $current_booking->id; ?>, false, false);
    $("#coupon").select2();
```

```
</script>
<?php include './layouts/footer.php'; ?>
```

View Bill

```
<?php
include './layouts/header.php';

$invoice_id = isset($_GET['bill']) ? $_GET['bill'] : false;
$invoice = Invoice::find_by_id($invoice_id);
$invoice->init_members();
$orders = Order_booking::find_completed_for_booking($invoice-
>bookingObj)["response"];
?>


<style>
   h4.row.header span{
      font-weight: 700;
   }
   #order_time,#order_id,.bolder{
      font-weight: 600;
   }

   article.table{
      padding: 1em;
   }

   .table_heading{
      background: #f9f9f9;
      padding-top: 16px;
      padding-bottom: 8px;
      box-shadow: 0px 2px 2px rgba(0,0,0,0.05);
   }

   .bigger{
      font-size: 1.3em;
   }

   div.table.order_contents{
      background: #fcfcfc;
      margin-top: 8px;
   }

   .order_item{
      padding: 16px;
```

```
    }

    #invoice_header{
        margin-top: 1em;
        padding: 2em;
        background: #a97e13;
        color: white;
    }

    #invoice_header h2{
        margin-bottom: 1em;
        padding-bottom: 0.5em;
        border-bottom: 1px solid #e2a91b;
    }

    #final_settlement{
        padding: 2em;
        border: 2px dashed darkgoldenrod;

    }
    #final_settlement span.row:first-child, #final_settlement span.row:last-child{
        color: darkgoldenrod;
        font-weight: 800;
        font-size: 1.3em;
        font-family: serif;
    }
    #final_settlement span.row:first-child *{
        padding: 1.5em;
        border-bottom: 1px solid darkgoldenrod;
    }
    #final_settlement span.row:last-child *{
        padding: 1.5em;
        border-top: 1px solid darkgoldenrod;
    }

    button#printer{
        position: fixed;
        right: 48px;
        top: 96px;
    }


</style>

<button onclick="printContent();" id="printer">
```

```
   Print Invoice Copy
</button>

<div class="container" id="invoice">

   <div class="col-md-offset-1 col-md-10">

      <article id="invoice_header">
         <h2 class="row"><?= $invoice->name(); ?></h2>
         <h3 class="row">
            <span class="col-md-6 text-right">Billed</span> <span
class="col-md-6 text-right">₹ <?= $invoice->total_amount; ?>/-</span>
         </h3>
      </article>

      <article class="table table-bordered table-hover table-striped">

         <h4 class="row">
            <span class="col-md-2">Order Id</span>
            <span class="col-md-10">Order On</span>
         </h4>
         <span class="row text-right">Orders placed during the booking
period</span>
         <?php while ($order = current($orders)): ?>
            <div class="row table_heading">
               <span class="col-md-2" id="order_id">
                  <?= $order->id ?>
               </span>
               <span class="col-md-10" id="order_time">
                  <?= DatabaseObject::format_datetime("d M Y h:i a", $order-
>date) ?>
               </span>
            </div>
            <div class="col-md-offset-2">
               <div class="table order_contents">
                  <?php
                  $contents = $order->contents;
                  $first_row = true;
                  while ($content = current($contents)):
                     ?>
                     <?php
                     if ($first_row):
                        ?>
                        <div class="row table_heading">
                           <strong class="col-md-1">Quantity</strong>
```

```
                        <strong class="col-md-6">Item</strong>
                        <strong class="col-md-3">Amount</strong>
                    </div>
                    <?php
                    $first_row = false;
                endif;
                ?>

                <div class="row order_item">
                    <span class="col-md-1"><?= $content->quantity;
?></span>
                    <span class="col-md-6"><?= $content->itemObj->item;
?></span>
                    <span class="col-md-3">₹ <?= $content->bill(); ?>/-
</span>
                </div>
                <?php
                next($contents);
            endwhile;
            ?>
        </div>
    </div>
    <?php
    next($orders);
endwhile;
?>
<section id="final_settlement">
    <span class="row">
        <span class="col-md-5">Invoice Status</span>
        <span class="col-md-5"><?= $invoice->status(); ?></span>
    </span>
    <span class="row">
        <span class="col-md-5">Stay at <?= $invoice->bookingObj-
>facilityObj->name(); ?></span>
        <span class="col-md-5"><em>₹ <?= $invoice->bookingObj-
>facilityObj->charges ?>/-</em> per day.</span>
    </span>
    <span class="row">
        <span class="col-md-5">Your stay was for</span>
        <span class="col-md-5"><?= $invoice->bookingObj-
>stay_period(); ?> days.</span>
    </span>

    <span class="row">
        <span class="col-md-5">Total amount for the Facility</span>
```

```
                <span class="col-md-5"><strong>₹ <?= $invoice->bookingObj->charges(); ?>/-</strong></span>
            </span>


            <big class="row">
                <span class="col-md-5">Total amount for the Booking <br>(<small>including Facility and Orders</small>)</span>
                <span class="col-md-5">₹ <?= $invoice->total_amount; ?>/-</span>
            </big>


            <span class="row">
                <span class="col-md-5">Coupon Applied</span>
                <span class="col-md-5"><?= $invoice->discountObj->name(); ?></span>
            </span>
            <span class="row">
                <span class="col-md-5">Amount Paid after Discounts</span>
                <span class="col-md-5">₹ <?= $invoice->amount_payable; ?>/-</span>
            </span>
        </section>
    </article>
  </div>
</div>

<script>
  function printContent() {
     $("#page_header").hide();
     $("#page_footer").hide();
     $("button#printer").hide();
     window.print();
     $("#page_header").show();
     $("#page_footer").show();
     $("button#printer").show();
  }
</script>

<?php include './layouts/footer.php'; ?>
```

Admin List Tables

```
<?php
```

```
$table = (isset($_GET["table"])) ? $_GET["table"] : "person";
$page_title = "Listing " . ucwords($table) . " table";

$limit = (isset($_GET['limit'])) ? $_GET['limit'] : 10;
$page = (isset($_GET['page'])) ? $_GET['page'] : 1;

$prev_page = ($page > 1) ? ($page - 1) : 1;
$next_page = $page + 1;

$nav_only = TRUE;
include './layouts/header.php';
admins_only();
?>
<div class="container-fluid">
   <?php
   global $database;
   if ($table) {
      $table_records = $table::find_limited($limit, $page);
   }
   ?>
   <nav class="navbar">
      <div class="navbar-header">
         <a class="navbar-toggle" data-toggle="collapse" data-
target="#table_list">
            <span class="glyphicon glyphicon-chevron-down"></span>
         </a>
         <a class="navbar-brand" href="#">Tables </a>
      </div>
      <div id="table_list" class="navbar-collapse collapse" data-spy="affix"
data-offset-top="300">
         <ul class="nav navbar-nav">
            <?php
            $tables = get_all_tables();
            while (($rec = current($tables)) !== FALSE):
               ?>
               <li class=" <?php if (strcasecmp($table, key($tables)) == 0) echo
"current active"; ?>">
                  <a href="?table=<?php echo key($tables); ?>"><?php echo
ucfirst(key($tables)); ?></a>
               </li>
               <?php
               next($tables);
            endwhile;
            ?>
         </ul>
```

```
    </div>
  </nav>

  <article id="details" class="panel panel-info">
    <h3 class="panel-heading"><?php echo ucfirst($table); ?></h3>
    <div class="panel-body">
      <a id="record-0"></a>
      <div class="table-responsive">
        <?php
        if ($table_records) {
          include '/layouts/table_render.php';
        } else {
          ?>
          <p class="text-danger">
          <big>No records found...</big>
          Try other tables or different page number.
          </p>
        </div>
      <?php }; ?>
    </div>

    <div class="panel-footer">
      <ul class="pager">
        <li class="previous"><a href="?table=<?= $table; ?>&page=<?=
$prev_page; ?>">Previous</a></li>
        <li class="next"><a href="?table=<?= $table; ?>&page=<?=
$next_page; ?>">Next</a></li>
      </ul>
    </div>

  </article>

  <?php
  $formFile = "./tableForms/{$table
      }_form.php";
  if (file_exists($formFile)) {
    include $formFile;
    $form = TRUE;
  } else {
    $form = FALSE;
    ?>
    <div class="container-fluid">
      <h4 class="text-danger">
        Could not find the Form for inserting new rows.
      </h4>
```

```
        </div>
        <?php
    }
    ?>


    <?php
    if (!$object->id) {
        $record = 1;
    } else {
        $record = $object->id;
    }
    ?>

    <a id="to_record" href="#record-<?= $record - 1; ?>" title="To the reocrd's
row">
        <span class="glyphicon glyphicon-menu-up"></span>
    </a>


</div>

<?php include './layouts/footer.php'; ?>

<?php if ($form): ?>
    <script>
        $("#form").validate(formRules);
        $("#form select").select2();
    <?php
    if ($object->id != '') {
        ?>
                $("html, body").animate({scrollTop: $("#form").parent().offset().top-
250}, 500);
                $("#form").prev().html("Update");
        <?php
    } else {
        ?>
                $("html, body").animate({scrollTop: $("#details").parent().offset().top},
500);
    <?php } ?>
        $("#form .row:last").append("<a class=\"btn btn-default\"
href=\"./list_tables.php?table=<?php echo $table; ?>\">Insert a new
Record</a>");
    </script>
<?php endif; ?>
```

Backend

Config

```
<?php

// Database Constants
defined('DB_SERVER') ? null : define("DB_SERVER", "localhost");
defined('DB_NAME') ? null : define("DB_NAME", "mannhotel");
defined('DB_USER') ? null : define("DB_USER", "root");
defined('DB_PASS') ? null : define("DB_PASS", "");


// Site Details
defined('SITE_TITLE') ? null : define('SITE_TITLE', "Mann Hotel");
defined('SITE_MOTO') ? null : define('SITE_MOTO', "A project for MCA");
defined('DEVELOPER_NAME') ? null : define('DEVELOPER_NAME', "Mandeep Kaur");
defined('DEVELOPER_INFO') ? null : define('DEVELOPER_INFO', "146438712");
defined('DEVELOPER_MAIL') ? null : define('DEVELOPER_MAIL',
"mandeepshabina@gmail.com");
defined('DEVELOPER_TEL') ? null : define('DEVELOPER_TEL', "+919041447146");
defined('SITE_CURRENCY') ? null : define('SITE_CURRENCY', "₹ ");
?>
```

Database

```
<?php

require_once(LIB_PATH . DS . "config.php");

class MySQLDatabase {

   private $connection;

   public function __construct() {
      $this->open_connection();
   }

   // Connection functions
   public function open_connection() {
      $this->connection = mysqli_connect(DB_SERVER, DB_USER, DB_PASS,
DB_NAME);
      if (!$this->connection) {
         die("Database connection failed: " .
               mysqli_connect_error() .
               " (" . mysqli_connect_errno() . ")"
         );
      }
```

```
}

public function close_connection() {
    if (isset($this->connection)) {
        mysqli_close($this->connection);
        unset($this->connection);
    }
}

// Database query functions
public function query($sql) {
    $result = mysqli_query($this->connection, $sql);
    $this->confirm_query($result, $sql);
    return $result;
}

private function confirm_query($result, $sql = "") {
    if (!$result) {
        echo '<pre>';
        debug_print_backtrace();
        echo '</pre>';
        die("<p>Database query <big>{$sql}</big> failed.</p>");
    }
}

public function escape_value($string) {
    $escaped_string = mysqli_real_escape_string($this->connection, $string);
    return $escaped_string;
}

// Database neutral functions
public function fetch_array($result_set) {
    return mysqli_fetch_array($result_set);
}

public function num_rows($result_set) {
    return mysqli_num_rows($result_set);
}

public function insert_id() {
    // get the last id inserted over the current database connection
    return mysqli_insert_id($this->connection);
}

public function affected_rows() {
```

```
        return mysqli_affected_rows($this->connection);
    }

}

$database = new MySQLDatabase();
$db = & $database;
?>
```

Database Object

```
<?php

//If it is going to need database, then it's probably smart to require it before we
start
require_once(LIB_PATH . DS . 'database.php');

class DatabaseObject {

    public static function find_by_sql($sql = "") {
        global $database;
        $result_set = $database->query($sql);
        $object_array = array();                // NOTE solution to problem #1
        while ($row = $database->fetch_array($result_set)) {
            $object_array[] = static::instantiate($row);      // NOTE using static::
instead of self:: for late static binding.
        }
        return $object_array;
    }

    public static function find_all() {
        return self::find_by_sql("SELECT * FROM " . static::$table_name);
    }

    public static function find_limited($limit = 10, $page = 1) {
        $offset = ($page - 1) * $limit;
        return self::find_by_sql("SELECT * FROM " . static::$table_name . " limit
$limit offset $offset");
    }

    public static function find_by_id($id) {
        if (empty($id)) {
            return false;
        }
        global $database;
        $id = $database->escape_value($id);
```

```
        $result_array = self::find_by_sql("SELECT * FROM " . static::$table_name . "
WHERE id = {$id} LIMIT 1");
        return !empty($result_array) ? array_shift($result_array) : false;
    }

    public static function count_all() {
        global $database;
        $sql = "SELECT COUNT(*) FROM " . static::$table_name;
        $result_set = $database->query($sql);
        $row = $database->fetch_array($result_set);
        return array_shift($row);
    }

    private static function instantiate($record) {
        $object = new static;
        foreach ($record as $attribute => $value) {
            if ($object->has_attribute($attribute)) {
                $object->$attribute = $value;
            }
        }
        return $object;
    }

    private function has_attribute($attribute) {
        $object_vars = $this->attributes();
        return array_key_exists($attribute, $object_vars); // all we need to confirm
is the key exists irrespective of the value.
    }

    public function attributes() {
        $attributes = array();
        foreach (static::$db_fields as $field) {
            if (property_exists($this, $field)) {
                $attributes[$field] = $this->$field;
            }
        }
        return $attributes;
    }

    protected function sanitized_attributes() {
        global $database;
        $clean_attributes = array();
        // sanitize the values before submitting
        // NOTE: does not alter the actual value of each attribute
```

```
    foreach ($this->attributes() as $key => $value) {
      $clean_attributes[$key] = $database->escape_value($value);
    }
    return $clean_attributes;
  }

  function insertion_attributes() {
    global $database;
    $clean_attributes = array();
    // sanitize the values before submitting
    // NOTE: does not alter the actual value of each attribute

    foreach ($this->attributes() as $key => $value) {
      if ($key != "id") {
        $clean_attributes[$key] = $database->escape_value($value);
      }
    }
    return $clean_attributes;
  }

  public function validate_attributes($attributes = array()) {
    foreach ($attributes as $attribute) {
      if ($this->has_attribute($attribute) && empty($this->$attribute)) {
        return false;
      }
    }
    return true;
  }

  public function show_attributes($attributes = array()) {
    $string = "<ul>";
    foreach ($attributes as $attribute) {
      $string .= "<strong>{$attribute}</strong> {$this->$attribute}";
      if ($this->has_attribute($attribute) && empty($this->$attribute)) {
        $string .= "<strong>{$attribute}</strong> {$this->$attribute}";
      }
    }
    $string .= "</ul>";
    return $string;
  }

  public function get_db_fields() {
    return static::$db_fields;
  }
```

```
  // CRUD Functions
  public function save() {
    // A new record won't have an id yet.
    $this->upload_dir();
    return (isset($this->id) || ($this->id != null)) ? $this->update() : $this-
>create();
  }

  public function upload_dir() {
    $upload_dir = UPLOAD_PATH . DS . static::$table_name;
    if (!is_dir($upload_dir)) {
      mkdir($upload_dir, 0777, true);
    }
    return $upload_dir;
  }

  public static function image_dir() {
    return 'uploads' . DS . static::$table_name;
  }

  public function upload_img($img) {
    $tmp_file = $img['tmp_name'];
    $name = basename($img['name']);
    $destination = time() . "-" . rand(1000, 9999) . "-" . $name;
    move_uploaded_file($tmp_file, $this->upload_dir() . DS . $destination);
    return $destination;
  }

  public function create() {
    global $database;
    $attributes = $this->insertion_attributes();
    $sql = "INSERT INTO " . static::$table_name . " (";
    $sql .= join(", ", array_keys($attributes));
    $sql .= ") VALUES (\"";
    $sql .= join("\", \"", array_values($attributes));
    $sql .= "\")";

    if ($database->query($sql)) {
      $this->id = $database->insert_id();
      return true;
    } else {
      return false;
    }
  }
```

```
    public function init_members() {
        // DO NOTHING;
    }

    public function table_edit($editable = FALSE) {
        global $current_user;

        if (method_exists($this, "init_members")) {
            $this->init_members();
        }

        if ($current_user->is_admin() && $editable) {
            return "<td id=\"record-{$this->id}\"  class=\"col-sm-12 col-md-
2\">"
                . "<form method=\"post\" action=\"./tableForms/delete.php\"
class=\"col-md-6\">"
                . "<button type=\"submit\" class=\"btn btn-small btn-danger\">"
                . "<span class=\"glyphicon glyphicon-trash\"></span>"
                . "</button>"
                . "<input type=\"hidden\" name=\"table_name\" value=\"" .
static::$table_name . "\"/>"
                . "<input type=\"hidden\" name=\"id\"value=\"" . $this->id .
"\"/>"
                . "<input type=\"hidden\" name=\"redirect_url\"value=\"" .
$_SERVER["REQUEST_URI"] . "\"/>"
                . "</form>"
                . "<div class=\"col-md-6\">"
                . "<a class=\"btn btn-warning\" href=\"?table=" .
static::$table_name . "&id=$this->id\">"
                . "<span class=\"glyphicon glyphicon-edit\"></span>" . $this->id
                . "</a>"
                . "</div>"
                . "</td>";
        } else {
            return "<td class=\"col-sm-12 col-md-2\">" . $this->id . "</td>";
        }
    }

    public function update() {
        global $database;
        $attributes = $this->sanitized_attributes();
        $attribute_pairs = array();
        foreach ($attributes as $key => $value) {
            $attribute_pairs[] = "{$key}='{$value}'";
        }
```

```
        $sql = "UPDATE " . static::$table_name . " SET ";
        $sql .= join(", ", $attribute_pairs);
        $sql .= " WHERE id=" . $database->escape_value($this->id);
        $database->query($sql);
        return ($database->affected_rows() == 1) ? true : false;
    }

    public function delete() {
        global $database;
        $sql = "DELETE FROM " . static::$table_name . " ";
        $sql .= "WHERE id=" . $database->escape_value($this->id) . " ";
        $sql .= "LIMIT 1";
        $database->query($sql);
        return ($database->affected_rows() == 1) ? true : false;
    }

    public function datetime($format = "h:i a, F d Y") {
        if ($this->datetime) {
            return static::format_datetime($format, $this->datetime);
        }
    }

    public static function format_datetime($format = "h:i a, F d Y", $datetime) {
        $dt = new DateTime($datetime);
        return $dt->format($format);
    }

    public static function form_date($date) {
        return static::format_datetime("Y-m-d", $date) . 'T' .
static::format_datetime("h:i:s", $date);
    }

    public function img() {
        return $this->image_dir() . DS . $this->img;
    }

    public function image_source() {
        return $this->image_dir() . DS . $this->img;
    }

    public function image() {
        return $this->avatar();
    }
```

```
    public function avatar($image_size = "72px", $class = "img img-thumbnail",
$tag_title = "-") {
        $title = ($tag_title == "-") ? "" : $this->name();
        $class = $class . " img-square img-" . str_replace("px", "", $image_size);
        if ($this->has_attribute('img')) {
            $startTag = "<img";
            $endTag = "/>";
            $content = " width=\"$image_size\""
                . " height=\"$image_size\""
                . " alt=\" " . $this->name() . "\""
                . " title=\" " . $title . "\""
                . " class=\"$class\""
                . " src=\"" . $this->img() . "\"";
        } else {
            $startTag = "<span>";
            $endTag = "</span>";
            $content = $this->name();
        }

        return $startTag
            . $content
            . $endTag;
    }

    public function intro($image_size = "72px", $class = "", $classImg = "img
img-thumbnail", $title = "org") {
        global $session;
        $user = $session->get_user_object();
        if ($user->is_admin()) {
            return "<a href=\"./list_tables.php?table=" . static::$table_name .
"&id=$this->id\" class=\"$class\" >"
                .
                $this->avatar($image_size, $classImg, $title)
                . "</a>";
        } else {
            return $this->avatar($image_size, $classImg, $title);
        }
    }

    public function title() {
        return $this->name();
    }

}
```

```
?>
```

Functions

```
<?php

// Required functions
function __autoload($class_name) {
   $class_name = strtolower($class_name);
   $path = LIB_PATH . DS . "{$class_name}.php";
   if (file_exists($path)) {
      require_once($path);
   } else {
      die("The file {$class_name}.php could not be found.");
   }
}

function redirect_to($location = NULL) {
   if ($location != NULL) {
      if (empty($location)) {
         $location = "index.php";
      }
      header("Location: {$location}");
      exit;
   }
}

// HTML Functions

function output_message($message = "", $class = "") {
   if (!empty($message)) {
      return "<p class=\"message {$class}\">{$message}</p>";
   } else {
      return "";
   }
}

function strip_zeros_from_date($marked_string = "") {
   // remove marked zeros
   $no_zeros = str_replace("*0", "", $marked_string);

   // remove remaining marks
   $cleaned_string = str_replace("*", "", $no_zeros);
   return $cleaned_string;
}
```

```
// LOG functions
function log_action($action, $message = "") {
   $file = SITE_ROOT . DS . 'logs' . DS . 'site_logs.txt';
   if ($handle = fopen($file, 'a')) {
      $log = "";
      $log .= strftime("%Y-%m-%d %H:%M:%S", time()) . " | ";
      $log .= $action . " " . $message . "\r\n";
      fwrite($handle, $log);
      fclose($handle);
   }
}

function get_all_logs() {
   $file = SITE_ROOT . DS . 'logs' . DS . 'site_logs.txt';
   $log = "";
   if ($handle = fopen($file, 'r')) {
      $log = fread($handle, filesize($file));
      fclose($handle);
   }
   $tempLog = trim($log);
   if (empty($tempLog)) {
      $log = "No log information available yet.";
   }
   return $log;
}

function wipe_all_logs($user_id) {
   $file = SITE_ROOT . DS . 'logs' . DS . 'site_logs.txt';

   $username = User::find_by_id($user_id)->username;
   if ($handle = fopen($file, 'w')) {
      $log = "user: {$username} with id: " . $user_id . " cleared all previous logs
on ";
      $log .= strftime("%Y-%m-%d %H:%M:%S", time());
      $log .= "\r\n";
      fwrite($handle, $log);
      fclose($handle);
   }
}

function admins_only() {
   give_access(TRUE);
}

function inside_persons_only() {
```

```
    give_access(FALSE, TRUE);
}

function members_only() {
    give_access(FALSE, FALSE);
}

function give_access($onlyAdmin = true, $onlyUsers = false) {
    global $session;
    $current_user = $session->get_user_object();

    $session->request_uri($_SERVER['REQUEST_URI']);
    if ($onlyUsers && $current_user->authority_level <= 1) {
        $session->message("Restricted Access.");
        redirect_to('index.php');
    }
    if ($onlyAdmin) {
        if (!$current_user->is_admin()) {
            $session->message("Only admins can view the requested page.");
            redirect_to('index.php');
        }
        // else keep going
    } else {
        if (!$session->is_logged_in()) {
            $session->message("You need to Login/Register first.");
            redirect_to('login.php');
        }
    }
}

function get_all_tables() {
    global $database;
    $result_set = $database->query("show tables");
    $database_tables = array();
    while ($table_detail = mysqli_fetch_array($result_set)) {
        $database_tables[$table_detail[0]] = get_table_details($table_detail[0]);
    }
    return $database_tables;
}

function get_table_details($table_name) {
    global $database;
    $table_result = $database->query("show columns from {$table_name}");
    while ($col = mysqli_fetch_assoc($table_result)) {
        $column[$col["Field"]] = $col["Type"];
```

```
    }
    return $column;
}

function get_input_type($column) {
    $col = strtoupper(substr($column, 0, 3));
    switch ($col) {
        case 'INT':
            return "number";
        case 'VAR':
            return 'text';
        case 'ENU':
            return 'select';

        case 'DAT':
        case 'TIM':
            return "datetime-local";

        default :
            return 'area';
    }
}

function create_input_element($name, $type, $default, $classes) {
    $column = get_input_type($type);
    if ($column === 'area') {
        $string = "<textarea id=\"{$name}\" name=\"{$name}\"
class=\"{$classes}\">{$default}</textarea>";
    } elseif ($column === 'select') {
        $optionsString = substr($type, strpos($type, '('), strpos($type, ')'));
        $optionsString = str_replace('(', '', $optionsString);
        $optionsString = str_replace(')', '', $optionsString);
        $values = explode(',', $optionsString);

        $string = "<select id=\"{$name}\" name=\"{$name}\"
class=\"{$classes}\">";
        foreach ($values as $option) {
            $option = str_replace('\'', '', $option);
            $string .= "<option value=\"{$option}\">{$option}</option>";
        }
        $string .= "</select>";
    } else {
        $string = "<input id=\"{$name}\" name=\"{$name}\" class=\"{$classes}\"
type=\"{$column}\" ";
        if ($name == "id") {
```

```
            $string .= " disabled ";
        } else {
            $string .= " value=\"{$default}\" ";
        }
        $string .= "/>";
    }
    return $string;
}

function form_date_time(){
    return date("Y-m-d") . 'T' . date("h:i:s");
}
?>
```

Session

```
<?php

/*
  A class to work with SESSIONs
  In our case, primarily to manage logging users in and out

  Keep in mind when working with sessions that it is generally inadvisable to
store DB-related objects in sessions.
 */

class Session {

    private $logged_in = false;
    public $user_id;
    public $user_type;
    public $message;
    public $request_uri;

    function __construct() {
        session_start();
        $this->check_message();
        $this->check_login();
        $this->check_request_uri();

        if ($this->logged_in) {
            // TODO actions to take right away if user is logged in
        } else {
            // TODO actions to take right away if user is not logged in
        }
```

```
}

public function is_logged_in() {
   return $this->logged_in;
}

public function login($user) {
   // database should find user based on username/password
   if ($user) {
      $this->user_id = $_SESSION['user_id'] = $user->id;
      $this->logged_in = true;
   }
}

public function logout() {
   unset($_SESSION['user_id']);
   unset($this->user_id);
   $this->logged_in = false;
}

private function check_login() {
   if (isset($_SESSION['user_id'])) {
      $this->user_id = $_SESSION['user_id'];
      $this->logged_in = true;
   } else {
      unset($this->user_id);
      $this->logged_in = false;
   }
}

private function check_message() {
   // Is there a message stored in the session?
   if (isset($_SESSION['message'])) {
      // Add it as an attribute and erase the stored version
      $this->message = $_SESSION['message'];
      unset($_SESSION['message']);
   } else {
      $this->message = "";
   }
}

private function check_request_uri() {
   if (isset($_SESSION['request_uri'])) {
      $this->request_uri = $_SESSION['request_uri'];
      unset($_SESSION['request_uri']);
```

```
        } else {
            $this->request_uri = "";
        }
    }

    public function message($msg = "") {
        if (!empty($msg)) {
            // then this is "set message"
            // make sure you understand why $this->message = $msg would not
work
            $_SESSION['message'] = $msg;
        } else {
            // then this is 'get message'
            return $this->message;
        }
    }

    public function current_booking($booking = "") {
        if (!empty($booking)) {
            $_SESSION['booking'] = $booking;
        } else {
            // then this is 'get message'
            return $_SESSION['booking'];
        }
    }

    public function request_uri($uri = "") {
        if (!empty($uri)) {
            $_SESSION['request_uri'] = $uri;
        } else {
            // then this is 'get message'
            return $this->request_uri;
        }
    }

    public function get_user_object() {
        if (isset($this->user_id)) {
            $user = Account::find_by_id($this->user_id);
            $user->init_members();
            return $user;
        } else {
            return new Account();
        }
    }
```

```
}

$session = new Session();
$message = $session->message();
$request_uri = $session->request_uri();
?>
```

Person

```
<?php

// If it is going to need the database, then it is probably smart to require it
before we start.
require_once(LIB_PATH . DS . "database.php");

class Person extends DatabaseObject {

   protected static $table_name = "person";
   protected static $db_fields = array('id', 'first_name', 'last_name', 'address',
'phone', 'email', 'dob', 'anniversary', 'img');
   public $id;
   public $first_name;
   public $last_name;
   public $address;
   public $phone;
   public $email;
   public $dob;
   public $anniversary;
   public $img = "";

   public static function make($first_name, $last_name, $phone, $email) {
      $person = new Person();
      $person->first_name = $first_name;
      $person->last_name = $last_name;
      $person->phone = $phone;
      $person->email = $email;
      return $person;
   }

   public function create_blank() {
      global $database;
      $sql = "INSERT INTO " . static::$table_name . " (";
      $sql .= "first_name, last_name, phone, email";
      $sql .= ") VALUES "
            . "("
            . "\"" . $this->first_name . "\","
```

```
            . "\"" . $this->last_name . "\","
            . "\"" . $this->phone . "\","
            . "\"" . $this->email . "\""
            . ")";

    if ($database->query($sql)) {
        $this->id = $database->insert_id();
        return true;
    } else {
        return false;
    }
}

public function save() {
    if (!$this->anniversary) {
        $this->anniversary = "0001-01-01";
    }
    parent::save();
}

public function name() {
    return $this->first_name . " " . $this->last_name;
}

public function renderTableHeader() {
    return '
        <thead>
            <tr class="row">
                <th class="col-sm-12 col-md-2 ">
                    Id
                </th>
                <th class="col-sm-12 col-md-2 ">
                    Image
                </th>
                <th class="col-sm-12 col-md-2 ">
                    First Name
                </th>
                <th class="col-sm-12 col-md-2 ">
                    Last Name
                </th>
                <th class="col-sm-12 col-md-2 ">
                    Address
                </th>
                <th class="col-sm-12 col-md-2 ">
                    Phone
```

```
            </th>
            <th class="col-sm-12 col-md-2 ">
              email
            </th>
            <th class="col-sm-12 col-md-2 ">
              DOB
            </th>
            <th class="col-sm-12 col-md-2 ">
              Anniversary
            </th>
          </tr>
        </thead>';
  }

  public function renderTableRow($edit) {
    return "
      <tr class=\"row\">"
        . $this->table_edit($edit)
        . "<td class=\"col-sm-12 col-md-2\">" . $this->avatar() . "</td>"
        . "<td class=\"col-sm-12 col-md-2\">" . $this->first_name .
"</td>"
        . "<td class=\"col-sm-12 col-md-2\">" . $this->last_name . "</td>"
        . "<td class=\"col-sm-12 col-md-2\">" . $this->address . "</td>"
        . "<td class=\"col-sm-12 col-md-2\">" . $this->phone . "</td>"
        . "<td class=\"col-sm-12 col-md-2\">" . $this->email . "</td>"
        . "<td class=\"col-sm-12 col-md-2\">" . $this->dob . "</td>"
        . "<td class=\"col-sm-12 col-md-2\">" . $this->anniversary .
"</td>"
        . "</tr>";
  }
}
```

Account

```
<?php

// If it is going to need the database, then it is probably smart to require it
before we start.
require_once(LIB_PATH . DS . "database.php");

class Account extends DatabaseObject {

  protected static $table_name = "account";
  public $id;
  public $user_name;
  public $password;
```

```
    public $authority_level;
    public $person;
    public $personObj;
    protected static $db_fields = array('id', 'user_name', 'password',
'authority_level', 'person');

    public static function make($user_name, $person, $password) {
        $account = new Account();
        $account->user_name = $user_name;
        $account->authority_level = 1;
        $account->person = $person;
        $account->password = $password;
        return $account;
    }

    public function is_admin() {
        if ($this->authority_level >= 2) {
            return TRUE;
        } else {
            return FALSE;
        }
    }

    public static function find_admins($limit) {
        return self::find_by_sql("SELECT * FROM "
                    . static::$table_name
                    . " where"
                    . " authority_level >=2"
                    . " order by authority_level desc"
                    . " limit $limit");
    }

    public function init_members() {
        if (!$this->personObj && isset($this->person)) {
            $this->personObj = Person::find_by_id($this->person);
        }
    }

    public function name() {
        $this->init_members();
        return $this->personObj->name();
    }

    public function title() {
        return "Account #" . $this->id . ", for " . $this->name();
```

```
    }

    public function avatar($image_size = "72px", $class = "img img-thumbnail",
$title = "org") {
        $this->init_members();
        return $this->personObj->avatar($image_size, $class, $title);
    }

    public function renderTableHeader() {
        return '
            <thead>
                <tr class="row">
                    <th class="col-sm-12 col-md-2">
                        Id
                    </th>
                    <th class="col-sm-12 col-md-2">
                        User Name
                    </th>
                    <th class="col-sm-12 col-md-2">
                        Password
                    </th>
                    <th class="col-sm-12 col-md-2">
                        Authority Level
                    </th>
                    <th class="col-sm-12 col-md-2">
                        Person
                    </th>
                    </thead>';
    }

    public function renderTableRow($edit) {
        $this->init_members();
        return "
            <tr class=\"row\">"
                . $this->table_edit($edit)
                . "<td class=\"col-sm-12 col-md-2\">" . $this->user_name .
"</td>"
                . "<td class=\"col-sm-12 col-md-2\">" . $this->password . "</td>"
                . "<td class=\"col-sm-12 col-md-2\">" . $this->authority_level .
"</td>"
                . "<td class=\"col-sm-12 col-md-2\">" . $this->personObj->intro()
. "</td>"
                . "</tr>";
    }
```

```
}

?>
```

Facility

```
<?php

// If it is going to need the database, then it is probably smart to require it
before we start.
require_once(LIB_PATH . DS . "database.php");

class Facility extends DatabaseObject {

   protected static $table_name = "facility";
   protected static $db_fields = array('id', 'title', 'type', 'number', 'floor',
'capacity', 'available', 'img','charges');
   public $id;
   public $title;
   public $type;
   public $number;
   public $floor;
   public $capacity;
   public $available;
   public $img;
   public $charges;

   public static function find_types() {
      global $database;
      $results = $database->query("select distinct type from facility");
      $response = array();
      while ($row = $database->fetch_array($results)) {
         $response[] = $row[0];
      }
      return $response;
   }

   public function name() {
      return $this->title;
   }

   public function description() {
      return "$this->title is a $this->type located at $this->floor.";
   }

   public static function show_case() {
```

```
        $sql = "select * from facility where id in (select max(id) from facility group
by type)";
        return self::find_by_sql($sql);
    }

    public static function find_available($capacity, $type, $check_in, $check_out,
$limit = 1) {
        $sql = "SELECT * FROM facility
                WHERE
                    id NOT IN
                    (SELECT facility FROM booking
                        WHERE
                            check_in >= STR_TO_DATE('{$check_in}', '%m-%d-%y')
                                AND check_out <= STR_TO_DATE('{$check_out}', '%m-%d-
%y'))";

        $sql .= " and"
                . " type = '$type'"
                . " and"
                . " capacity >= $capacity"
                . " order by capacity asc"
                . " limit $limit";
        return Facility::find_by_sql($sql);
    }

    public function renderTableHeader() {
        return '
            <thead>
                <tr class="row">
                    <th class="col-sm-12 col-md-2 ">
                        Id
                    </th>
                    <th class="col-sm-12 col-md-2 ">
                        Title
                    </th>
                    <th class="col-sm-12 col-md-2 ">
                        Type
                    </th>
                    <th class="col-sm-12 col-md-2 ">
                        Number
                    </th>
                    <th class="col-sm-12 col-md-2 ">
                        Floor
                    </th>
                    <th class="col-sm-12 col-md-2 ">
```

```
                Capacity
              </th>
              <th class="col-sm-12 col-md-2 ">
                Available
              </th>
              <th class="col-sm-12 col-md-2 ">
                Charges
              </th>
              <th class="col-sm-12 col-md-2 ">
                Image
              </th>
            </tr>
          </thead>';
    }

    public function renderTableRow($edit) {
        return "
            <tr class=\"row\">"
              . $this->table_edit($edit)
              . "<td class=\"col-sm-12 col-md-2\">" . $this->title . "</td>"
              . "<td class=\"col-sm-12 col-md-2\">" . $this->type . "</td>"
              . "<td class=\"col-sm-12 col-md-2\">" . $this->number . "</td>"
              . "<td class=\"col-sm-12 col-md-2\">" . $this->floor . "</td>"
              . "<td class=\"col-sm-12 col-md-2\">" . $this->capacity . "</td>"
              . "<td class=\"col-sm-12 col-md-2\">" . $this->available . "</td>"
              . "<td class=\"col-sm-12 col-md-2\">" . $this->charges . "</td>"
              . "<td class=\"col-sm-12 col-md-2\">" . $this->avatar() . "</td>"
              . "</tr>";
    }

}

?>
```

Booking

```
<?php

// If it is going to need the database, then it is probably smart to require it
before we start.
require_once(LIB_PATH . DS . "database.php");

class Booking extends DatabaseObject {

    protected static $table_name = "booking";
```

```
    protected static $db_fields = array('id', 'account', 'check_in', 'check_out',
'facility');
    public $id;
    public $account;
    public $check_in;
    public $check_out;
    public $facility;
    public $facilityObj;
    public $accountObj;
    public $orders;
    public $bill;
    public $invoiceObj;

    public static function make($account, $facility, $check_in, $check_out) {
        $booking = new Booking();
        $booking->account = $account;
        $booking->facility = $facility;
        $booking->check_in = $check_in;
        $booking->check_out = $check_out;
        return $booking;
    }

    public function init_members() {
        if (!$this->accountObj && isset($this->account)) {
            $this->accountObj = Account::find_by_id($this->account);
            $this->accountObj->init_members();
        }
        if (!$this->facilityObj && isset($this->facility)) {
            $this->facilityObj = Facility::find_by_id($this->facility);
        }
        if (!$this->orders) {
            $orders = Order_booking::find_for_booking($this);
            $this->orders = $orders['response'];
            $this->bill = $orders['bill'];
        }

        if (!$this->invoiceObj) {
            $this->invoiceObj = Invoice::find_for_booking($this);
            if ($this->invoiceObj instanceof Invoice && isset($this->invoiceObj-
>ammount_payable)) {
                $this->bill = $this->invoiceObj->amount();
            }
        }
    }
```

```
    public function charges() {
        return $this->stay_period() * $this->facilityObj->charges;
    }

    public function stay_period($format = '%a') {
        $this->init_members();

        if ($this->invoiceObj instanceof Invoice) {
            $datetime2 = new DateTime($this->check_out);
        } else {
            $datetime2 = new DateTime();
        }
        $datetime1 = new DateTime($this->check_in);
        $interval = $datetime1->diff($datetime2);
        return $interval->format($format);
    }

    public function name() {
        $this->init_members();
        return "Booking #{$this->id} by " . $this->accountObj->name(); //. "
<small>\"" . $this->check_in() . " to " . $this->check_out() . "\"</small>";
    }

    public function bill() {
        $re = 0;
        $msg = "Not Generated yet ";
        if ($this->invoiceObj instanceof Invoice) {
            $re = $this->invoiceObj->amount();
            $msg = "You paid ";
        } else {
            $re = $this->bill;
            $msg = "Estimate Bill ";
        }
        $res = "{$msg}: ₹ {$re} /-";
        return $res;
    }

    public function past_checkout() {
        $this->init_members();
        $today = strtotime('now');
        $check_out = strtotime($this->check_out);
        $datediff = $today - $check_out;
        $days = floor($datediff / (60 * 60 * 24));
        if ($days > 1) {
            return TRUE;
```

```
    } else {
        return FALSE;
    }
}

public function title() {
    return $this->check_in() . " to " . $this->check_out();
}

public function check_in($format = "h:i a, F d Y") {
    return DatabaseObject::format_datetime($format, $this->check_in);
}

public function check_out($format = "h:i a, F d Y") {
    return DatabaseObject::format_datetime($format, $this->check_out);
}

public static function find_all_for_account($account) {
    $sql = "select * from "
          . static::$table_name
          . " where account=$account order by check_in desc, check_out desc";
    return Booking::find_by_sql($sql);
}

public function settle() {
    if ($this->id) {
        $sqlFailed = "UPDATE order_contents"
              . " set"
              . " order_contents.status ='Failed'"
              . " where"
              . " order_contents.order_id in"
              . " ("
              . "select id from order_booking"
              . " WHERE"
              . " booking = $this->id"
              . ")"
              . " and"
              . " status in ('Booked','Failed')";
        $sqlDelivered = "UPDATE order_contents"
              . " set"
              . " order_contents.status ='Delivered'"
              . " where"
              . " order_contents.order_id in"
              . " ("
              . "select id from order_booking"
```

```
                . " WHERE"
                . " booking = $this->id"
                . ")"
                . " and"
                . " status not in ('Booked','Failed')";

        global $database;
        $database->query($sqlFailed);
        $database->query($sqlDelivered);
    }
}

public function renderTableHeader() {
    return '
        <thead>
            <tr class="row">
                <th class="col-sm-12 col-md-2 ">
                    Id
                </th>
                <th class="col-sm-12 col-md-2 ">
                    Account
                </th>
                <th class="col-sm-12 col-md-2 ">
                    Check In
                </th>
                <th class="col-sm-12 col-md-2 ">
                    Check Out
                </th>
                <th class="col-sm-12 col-md-2 ">
                    Facility
                </th>
            </tr>
        </thead>';
}

public function renderTableRow($edit) {
    return "
        <tr class=\"row\">"
            . $this->table_edit($edit)
            . "<td class=\"col-sm-12 col-md-2\">" . $this->accountObj-
>intro() . "</td>"
            . "<td class=\"col-sm-12 col-md-2\">" . $this->check_in() . "</td>"
            . "<td class=\"col-sm-12 col-md-2\">" . $this->check_out() .
"</td>"
```

```
            . "<td class=\"col-sm-12 col-md-2\">" . $this->facilityObj->intro()
. "</td>"
            . "</tr>";
   }

}

?>
```

Menu

```
<?php

// If it is going to need the database, then it is probably smart to require it
before we start.
require_once(LIB_PATH . DS . "database.php");

class Menu extends DatabaseObject {

   protected static $table_name = "menu";
   protected static $db_fields = array('id', 'item', 'price', 'hotel', 'description',
'img');
   public $id;
   public $item;
   public $price;
   public $hotel;
   public $description;
   public $img;
   public $hotelObj;

   public function init_members() {
      if (!$this->hotelObj && isset($this->hotel)) {
         $this->hotelObj = Hotel::find_by_id($this->hotel);
      }
   }

   public function name() {
      return $this->item . " " . $this->title();
   }

   public function title() {
      return "₹" . $this->price ."/-";
   }

   public static function find_expensive($limit = 10) {
```

```
        return self::find_by_sql("select * from menu order by price desc limit
$limit");
    }

    public function renderTableHeader() {
        return '
            <thead>
                <tr class="row">
                    <th class="col-sm-12 col-md-2 ">
                        Id
                    </th>
                    <th class="col-sm-12 col-md-2 ">
                        Item
                    </th>
                    <th class="col-sm-12 col-md-2 ">
                        Price
                    </th>
                    '
//                  <th class="col-sm-12 col-md-2 ">
//                      Hotel
//                  </th>
              . '<th class="col-sm-12 col-md-2 ">
                        Description
                    </th>
                    <th class="col-sm-12 col-md-2 ">
                        Image
                    </th>
                </tr>
            </thead>';
    }

    public function renderTableRow($edit) {
        $this->init_members();
        return "
            <tr class=\"row\">"
                . $this->table_edit($edit)
                . "<td class=\"col-sm-12 col-md-2\">" . $this->item . "</td>"
                . "<td class=\"col-sm-12 col-md-2\">" . $this->price . "</td>"
//                . "<td class=\"col-sm-12 col-md-2\">" . $this->hotelObj->intro()
. "</td>"
                . "<td class=\"col-sm-12 col-md-2\">" . $this->description .
"</td>"
                . "<td class=\"col-sm-12 col-md-2\">" . $this->avatar() . "</td>"
                . "</tr>";
    }
```

```
}

?>
```

Order Booking

```
<?php

// If it is going to need the database, then it is probably smart to require it
before we start.
require_once(LIB_PATH . DS . "database.php");

class Order_booking extends DatabaseObject {

   protected static $table_name = "order_booking";
   protected static $db_fields = array('id', 'account', 'booking', 'date');
   public $id;
   public $account;
   public $booking;
   public $date;
   public $accountObj, $bookingObj;
   public $contents;
   public $bill;
   public $final_check;

   public static function make($account, $booking) {
      $obj = new Order_booking();
      $obj->account = $account;
      $obj->booking = $booking;
      return $obj;
   }

   public function create_new() {
      global $database;
      $sql = "INSERT INTO " . static::$table_name . " (";
      $sql .= "account,booking";
      $sql .= ") VALUES (";
      $sql .= "$this->account,$this->booking";
      $sql .= ")";
      if ($database->query($sql)) {
         $this->id = $database->insert_id();
         return true;
      } else {
         return false;
      }
```

```
}

public static function find_completed_for_booking($booking) {
   return Order_booking::find_sq($booking, TRUE);
}

public static function find_for_booking($booking) {
   return Order_booking::find_sq($booking, FALSE);
}

private static function find_sq($booking, $onlyDelivered) {
   if (!$booking->id) {
      return false;
   }
   $sql = "select * from " . static::$table_name
         . " where booking={$booking->id} order by id desc,date desc";
   $db_rows = self::find_by_sql($sql);
   $response = array();
   $bill = 0;
   while ($row = current($db_rows)) {
      $row->bookingObj = $booking;
      $row->final_check = $onlyDelivered;
      $row->init_members();
      $response[] = $row;
      $bill += $row->bill;
      next($db_rows);
   }

   $lak["response"] = $response;
   $lak["bill"] = $bill;
   return $lak;
}

public function init_members() {
   if (!$this->accountObj && isset($this->account)) {
      $this->accountObj = Account::find_by_id($this->account);
      $this->accountObj->init_members();
   }
   if (!$this->bookingObj && isset($this->booking)) {
      $this->bookingObj = Booking::find_by_id($this->booking);
   }

   if (!$this->contents) {
      if ($this->final_check) {
         $response_contents = Order_contents::find_delivered($this);
```

```
        } else {
            $response_contents = Order_contents::find_for_order($this);
        }
        $this->contents = $response_contents["orders"];
        $this->bill = $response_contents["bill"];
    }
}

public function name() {
    $this->init_members();
    return "Order #{$this->id}"; //." for " . $this->bookingObj->name();
}

public function renderTableHeader() {
    return '
        <thead>
            <tr class="row">
                <th class="col-sm-12 col-md-2 ">
                    Id
                </th>
                <th class="col-sm-12 col-md-2 ">
                    Account
                </th>
                <th class="col-sm-12 col-md-2 ">
                    Booking
                </th>
                <th class="col-sm-12 col-md-2 ">
                    Date
                </th>
            </tr>
        </thead>';
}

public function renderTableRow($edit) {
    $this->init_members();
    return "
        <tr class=\"row\">"
            . $this->table_edit($edit)
            . "<td class=\"col-sm-12 col-md-2\">" . $this->accountObj-
>intro() . "</td>"
            . "<td class=\"col-sm-12 col-md-2\">" . $this->bookingObj-
>intro() . "</td>"
            . "<td class=\"col-sm-12 col-md-2\">" . $this->date . "</td>"
            . "</tr>";
}
```

```
}

?>
```

Order Contents

```
<?php

// If it is going to need the database, then it is probably smart to require it
before we start.
require_once(LIB_PATH . DS . "database.php");

class Order_contents extends DatabaseObject {

   protected static $table_name = "order_contents";
   protected static $db_fields = array('id', 'order_id', 'menu_item', 'quantity',
'status');

   const ORDER_BOOKED = 'Booked';
   const ORDER_UNDER_PROCESS = 'Under Process';
   const ORDER_ON_THE_WAY = 'On the way';
   const ORDER_DELIVERED = 'Delivered';
   const ORDER_FAILED = 'Failed';

   public $id;
   public $order_id;
   public $menu_item;
   public $quantity;
   public $status;
   public $orderObj;
   public $itemObj;

   public function bill() {
      $this->init_members();
      return $this->quantity * $this->itemObj->price;
   }

   public static function make($order_id, $menu_item, $quantity) {
      $content = new Order_contents();
      $content->order_id = $order_id;
      $content->menu_item = $menu_item;
      $content->status = "Booked";
      $content->quantity = $quantity;
      return $content;
   }
```

```
    public function init_members() {
        if (!$this->itemObj && isset($this->menu_item)) {
            $this->itemObj = Menu::find_by_id($this->menu_item);
        }
        if (!$this->orderObj && isset($this->order_id)) {
            $this->orderObj = Order_booking::find_by_id($this->order_id);
        }
    }

    public static function find_for_order($order) {
        if (!$order->id) {
            return false;
        }
        $sql = "select * from " . static::$table_name
                . " where"
                . " order_id = {$order->id}";
        return Order_contents::find_served($sql, $order);
    }

    public static function find_delivered($order) {
        if (!$order->id) {
            return false;
        }
        $sql = "select * from " . static::$table_name
                . " where"
                . " order_id = {$order->id} and status = 'Delivered'";
        return Order_contents::find_served($sql, $order);
    }

    private static function find_served($sql, $order) {
        $mOrders = self::find_by_sql($sql);
        $response = array();
        $bill = 0;
        while ($content_object = current($mOrders)) {
            $content_object->orderObj = $order;
            $content_object->init_members();
            $response[] = $content_object;
            $bill += $content_object->itemObj->price * $content_object-
>quantity;
            next($mOrders);
        }
        $lak["orders"] = $response;
        $lak["bill"] = $bill;
        return $lak;
```

```
    }

    public function name() {
        $this->init_members();
        return $this->itemObj->intro() . " " . $this->itemObj->name();
//        return "Order for " . $this->itemObj->intro(); //. " in " . $this-
>orderObj->name();
    }

    public function item_class() {
        $status = strtoupper($this->status);

        switch ($status) {
            case static::ORDER_BOOKED:
                return "info";
            case static::ORDER_UNDER_PROCESS:
            case 'on the way':
                return "warning";
            case static::ORDER_DELIVERED:
                return "success";
            case static::ORDER_FAILED:
                return 'danger';
            default :
                return "default";
        }
    }

    public function renderTableHeader() {
        return '
            <thead>
                <tr class="row">
                    <th class="col-sm-12 col-md-2 ">
                        Id
                    </th>
                    <th class="col-sm-12 col-md-2 ">
                        Order Id
                    </th>
                    <th class="col-sm-12 col-md-2 ">
                        Menu Item
                    </th>
                    <th class="col-sm-12 col-md-2 ">
                        Quantity
                    </th>
                    <th class="col-sm-12 col-md-2 ">
                        Status
```

```
                </th>
            </tr>
        </thead>';
    }

    public function renderTableRow($edit) {
        $this->init_members();
        return "
            <tr class=\"row\">"
            . $this->table_edit($edit)
            . "<td class=\"col-sm-12 col-md-2\">" . $this->orderObj->intro() .
"</td>"
            . "<td class=\"col-sm-12 col-md-2\">" . $this->itemObj->intro() .
"</td>"
            . "<td class=\"col-sm-12 col-md-2\">" . $this->quantity . "</td>"
            . "<td class=\"col-sm-12 col-md-2\">" . $this->status . "</td>"
            . "</tr>";
    }

    public static function order_status() {
        $rs = array();
        $rs[] = static::ORDER_BOOKED;
        $rs[] = static::ORDER_UNDER_PROCESS;
        $rs[] = static::ORDER_ON_THE_WAY;
        $rs[] = static::ORDER_DELIVERED;
        $rs[] = static::ORDER_FAILED;

        return $rs;
    }

}
?>
```

Invoice

```
<?php

// If it is going to need the database, then it is probably smart to require it
before we start.
require_once(LIB_PATH . DS . "database.php");

class Invoice extends DatabaseObject {

    protected static $table_name = "invoice";
```

```
    protected static $db_fields = array('id', 'booking', 'generation_date',
'total_amount', 'discount', 'amount_payable', 'status');
    public $id;
    public $booking;
    public $generation_date;
    public $total_amount;
    public $discount;
    public $amount_payable;
    public $status;
    public $bookingObj;
    public $discountObj;

    public function name() {
        $this->init_members();
        return "Invoice #$this->id for " . $this->bookingObj->name();
    }

    public function init_members() {
        if (!$this->bookingObj && isset($this->booking)) {
            $this->bookingObj = Booking::find_by_id($this->booking);
            $this->bookingObj->init_members();
        }

        if (!$this->discountObj && isset($this->discount)) {
            $this->discountObj = Discount_coupons::find_by_id($this->discount);
            $this->discountObj->init_members();
        }
    }

    public function status(){
        switch (strtolower($this->status)){
            case "paid":
                return "Your bill has been fully paid.";

            case "over due":
                return "Please pay the amount at reception.";

            case "Waived":
                return "Your bill is on the house.";

            default :
                return "Your invoice is in $this->status stage." ;
        }
    }
```

```
    public function amount() {
        return "$this->amount_payable ($this->total_amount)";
    }

    public static function find_for_booking($booking) {
        if ($booking->id) {
            $sql = "select * from " . static::$table_name . " where
booking={$booking->id}";
            $nInvoice = Invoice::find_by_sql($sql);
            if ($nInvoice) {
                $nInvoice = array_shift($nInvoice);
                $nInvoice->bookingObj = $booking;
                $nInvoice->init_members();
                return $nInvoice;
            } else {
                return false;
            }
        } else {
            return false;
        }
    }

    public static function make_for_booking($booking) {
        $invoice = new Invoice();
        $invoice->booking = $booking->id;
        $orders_in = Order_booking::find_completed_for_booking($booking);
        $invoice->total_amount = $orders_in['bill'] + $booking->charges();
        $invoice->amount_payable = $orders_in['bill'] + $booking->charges();
        return $invoice;
    }

    public function generation_date($format = "h:i a, F d Y") {
        return DatabaseObject::format_datetime($format, $this-
>generation_date);
    }

    public function renderTableHeader() {
        return '
            <thead>
                <tr class="row">
                    <th class="col-sm-12 col-md-2 ">
                        Id
                    </th>
                    <th class="col-sm-12 col-md-2 ">
                        Booking
```

```
                </th>
                <th class="col-sm-12 col-md-2 ">
                   Generation Date
                </th>
                <th class="col-sm-12 col-md-2 ">
                   Total Amount
                </th>
                <th class="col-sm-12 col-md-2 ">
                   Discount
                </th>
                <th class="col-sm-12 col-md-2 ">
                   Amount Payable
                </th>
                <th class="col-sm-12 col-md-2 ">
                   Status
                </th>
                </tr>
          </thead>';
   }

   public function renderTableRow($edit) {
      $this->init_members();
      return "
         <tr class=\"row\">"
            . $this->table_edit($edit)
            . "<td class=\"col-sm-12 col-md-2\">" . $this->bookingObj-
>intro() . "</td>"
            . "<td class=\"col-sm-12 col-md-2\">" . $this->generation_date() .
"</td>"
            . "<td class=\"col-sm-12 col-md-2\">" . $this->total_amount .
"</td>"
            . "<td class=\"col-sm-12 col-md-2\">" . $this->discountObj-
>intro() . "</td>"
            . "<td class=\"col-sm-12 col-md-2\">" . $this->amount_payable .
"</td>"
            . "<td class=\"col-sm-12 col-md-2\">" . $this->status . "</td>"
            . "</tr>";
   }

}

?>
```

Initialize

```
<?php
```

```
// Web site constants
// Web site constants
defined('DS') ? null : define('DS', '\\');
//defined('SITE_ROOT') ? null : define('SITE_ROOT',
$_SERVER['DOCUMENT_ROOT']);
defined('SITE_ROOT') ? null : define('SITE_ROOT', "C:" . DS . "wamp64" . DS .
"www" . DS . "mannHotel");
defined('LIB_PATH') ? null : define('LIB_PATH', SITE_ROOT . DS . 'includes');
defined('UPLOAD_PATH') ? null : define('UPLOAD_PATH', SITE_ROOT . DS .
'uploads');

// load configuration file
require_once(LIB_PATH . DS . "config.php");

// load basic functions
require_once(LIB_PATH . DS . "functions.php");

// load core objects
require_once(LIB_PATH . DS . "session.php");
require_once(LIB_PATH . DS . "database.php");
require_once(LIB_PATH . DS . "database_object.php");

// load database-related classes
require_once(LIB_PATH . DS . "account.php");
require_once(LIB_PATH . DS . "booking.php");
require_once(LIB_PATH . DS . "facility.php");
require_once(LIB_PATH . DS . "hotel.php");
require_once(LIB_PATH . DS . "invoice.php");
require_once(LIB_PATH . DS . "menu.php");
require_once(LIB_PATH . DS . "order_booking.php");
require_once(LIB_PATH . DS . "order_contents.php");
require_once(LIB_PATH . DS . "person.php");

?>
```

Business Logic

Book Table

```
<?php
require_once('../includes/initialize.php');
global $session;
$current_user = $session->get_user_object();

if (!$current_user) {
   $first_name = $_POST['first_name'];
```

```
    $last_name = $_POST['last_name'];
    $phone = $_POST['phone_no'];
    $email = $_POST['email'];
    $password = $_POST['password'];
    $destination = time() . "_" . rand(1000, 9999);

    $user_name = $first_name . "_" . $last_name . "_" . $destination;

    $user_name = strtolower($user_name);

    $person = Person::make($first_name, $last_name, $phone, $email);
    $person->create_blank();

    $account = Account::make($user_name, $person->id, $password);
    $account->save();
    $account->init_members();
} else {
    $account = $current_user;
    $account->init_members();
    $person = $account->personObj;
}

$check_in = $_POST['check_in'];
$check_out = $_POST['check_out'];
$capacity = $_POST['capacity'];
$type = $_POST['type'];
$hotel = Hotel::find_by_id(1);

$available = Facility::find_available($capacity, $type, $check_in, $check_out);

$facility = array_shift($available);
$booking = Booking::make($account->id, $facility->id, $check_in,
$check_out);
echo $booking->save();
$redirect_url = (isset($_POST['redirect_url'])) ? $_POST['redirect_url'] :
"../my_bookings.php?booking={$booking->id}";
redirect_to($redirect_url);

echo '<pre>';
print_r($facility);
echo '</pre>';
```

Booking Details

```
<article class="panel">
    <?php
```

```
   require_once '../includes/initialize.php';

   $order_more = (isset($_POST['order_more']) &&
strtolower($_POST['order_more'] == 'true')) ? TRUE : FALSE;
   $link_checkout = (isset($_POST['link_checkout']) &&
strtolower($_POST['link_checkout'] == 'true')) ? TRUE : FALSE;
   $current_booking = Booking::find_by_id($_POST['id']);
   if ($current_booking):
      $session->current_booking($current_booking->id);
      $current_booking->init_members();
      $past_checkout = $current_booking->past_checkout();
      if ($past_checkout) {
         $order_more = FALSE;
      }
      ?>
      <h1 class="panel-heading">
         <?php echo $current_booking->name(); ?>
      </h1>
      <section class = "panel-body">
         <h4 class = "heading" ><?php echo $current_booking->check_in() . "
to " . $current_booking->check_out();
         ?></h4>
         <?php
         if ($current_booking->invoiceObj instanceof Invoice) {
            ?>
            <h4 class="subheading">
               ==
               <span class="text small simple">You checked out at</span>
<span class="text text-primary glyphicon glyphicon-calendar"></span>
               <strong>
                  <span class="text text-primary simple"><?php echo
$current_booking->invoiceObj->generation_date(); ?></span>
               </strong>
               ==
            </h4>

            <a target="_newWindow" href="view_bill.php?bill=<?=
$current_booking->invoiceObj->id; ?>" class="btn btn-primary btn-sm">View
Invoice</a>
            <?php
         } else {
            if ($link_checkout) {
               ?>

               <?php if ($past_checkout) {
```

```
                ?>
                <p class="alert alert-danger">
                    You should have checked out by now!
                </p>
                <?php
            }
            ?>
            <a href = "checkout.php" class = "btn btn-primary pad-8 mar-8">
                Checkout <span class = "glyphicon glyphicon-log-
out"></span>
            </a>

            <?php
        }
        if ($order_more) {
            ?>
            <button type="button" class="btn btn-danger" data-
toggle="modal" data-target="#newOrderForm_modal"
onclick="setTimeout(renderForm, 200);">
                <span class="glyphicon glyphicon-cutlery"></span> Order
Something
            </button>
            <?php
            include '../layouts/new_order.php';
        }
    }
    ?>


    <h4 class="subheading">Orders</h4>
    <a class="bill" href="#checkout">
        <small><span class="glyphicon glyphicon-
file"></span></small><?php echo $current_booking->bill(); ?>
    </a>
    <?php $orders = $current_booking->orders; ?>
    <ul class="list-group orders-list">
        <?php while ($order = current($orders)): ?>
            <li class="list-group-item list-group-item-heading">
                <h5>
                    <strong><?php echo $order->name(); ?></strong>
                </h5>
                <div class="bill">
                    <small><span class="glyphicon glyphicon-
cutlery"></span></small> ₹ <?php echo $order->bill; ?>/-
                </div>
```

```
                <ul class="list-group bookings">
                    <?php while ($menu_item = current($order->contents)): ?>
                        <li class="list-group-item list-group-item-<?php echo
$menu_item->item_class(); ?>">
                            <span>
                                <?php echo $menu_item->name(); ?>
                            </span>
                            Quantity:<strong> <?php echo $menu_item->quantity;
?></strong>
                            <?php
                            if (isset($_POST['editing']) && $_POST['editing'] == 'true'):
                                ?>
                                <select id="menu_order_<?= $menu_item->id; ?>"
onchange="change_order_status(this);">
                                    <?php
                                    $options = Order_contents::order_status();
                                    $selected_option = $menu_item->status;
                                    include '../layouts/options_list.php';
                                    ?>
                                </select>
                                <div style="display: inline-block;" id="text_<?=
$menu_item->id; ?>"></div>

                                <?php
                            else:
                                ?>
                                <small>Status: <em>(<?php echo $menu_item-
>status; ?></em>)</small>

                            <?php endif; ?>
                        </li>
                        <?php
                        next($order->contents);
                    endwhile;
                    ?>
                </ul>
            </li>
            <?php
            next($orders);
        endwhile;
        ?>
    </ul>
</section>
<?php else: ?>
    <h1>Select a valid booking....</h1>
```

```
   <?php endif; ?>
</article>


<script>

   var current_or;
   var order_id;
   function change_order_status(order) {
      current_or = order.id;
      order_id = current_or.substr(current_or.lastIndexOf("_") + 1,
current_or.length);
      url = "./logic/update_order.php";
      data = {
         content_id: order_id,
         content_status: $("#" + current_or).val()
      };

      $.post(url, data, function (response) {
         console.log(response.updated);
         console.log(response);
         var span = "#text_" + order_id;
         console.log(span);
         var html;
         if (response.updated) {
            html = "<span class=\"text-success\">Updated " + order_id +
"</span>";
            $(span).html(html);
         } else {
            html = "<span class=\"text-danger\">Failed" + order_id +
"</span>";
            $(span).html(html);
         }
      }, "json");


   }
</script>
```

Checkout

```
<?php

require_once './../includes/initialize.php';
$cpn = (isset($_POST['coupon_id'])) ? $_POST['coupon_id'] : FALSE;
$bkng = (isset($_POST['booking_id'])) ? $_POST['booking_id'] : FALSE;
```

```
$amnt = (isset($_POST['amount'])) ? $_POST['amount'] : FALSE;

$response = array();
$response['error'] = TRUE;
$response['data'] = array();
if ($cpn && $bkng && $amnt) {
   $current_booking = Booking::find_by_id($bkng);
   $current_booking->settle();
   $invoice = Invoice::make_for_booking($current_booking);

   $coupon = Discount_coupons::find_by_id($cpn);
   $bill = $coupon->process($amnt);

   $invoice->generation_date = form_date_time();
   $invoice->discount = $cpn;
   $invoice->amount_payable = $bill;
   $invoice->status = "Paid";
   $invoice->save();

   $current_booking->check_out = $invoice->generation_date;
   $current_booking->save();
   $response['bill'] = $bill;
   $response['error'] = FALSE;
   $response['data'] = $invoice;

   $url = "../my_booking.php?booking=$current_booking->id";
   redirect_to($url);
} else {
   $response['error'] = TRUE;
   $response['msg'] = 'There was a problem processing your checkout.';
}
?>
<?php print_r($response); ?>
```

Coupon Validations

```
<?php

require_once './../includes/initialize.php';
$cpn = (isset($_POST['coupon'])) ? $_POST['coupon'] : FALSE;
$bkng = (isset($_POST['booking'])) ? $_POST['booking'] : FALSE;
$amnt = (isset($_POST['amount'])) ? $_POST['amount'] : FALSE;

log_action("cpn",$cpn);
log_action("bkng",$bkng);
log_action("amnt",$amnt);
```

```
$response = array();
$response['error'] = TRUE;
$response['data'] = array();
if ($cpn && $bkng && $amnt) {
   $coupon = Discount_coupons::find_by_id($cpn);
   $bill = $coupon->process($amnt);
   $response['bill'] = $bill;
   $response['error'] = FALSE;

   if ($amnt == $bill) {
      $response['error'] = TRUE;
      $response['msg'] = "You are not eligible for this coupon.";
   }

} else {
   $response['error'] = TRUE;
   $response['msg'] = 'There was a problem processing your coupon.';
}

log_action("aa", json_encode($_POST));
log_action("aa", json_encode($response));
echo json_encode($response);
```

Order

```
<?php
require_once '../includes/initialize.php';
$current_user = $session->get_user_object();
$booking = $_POST['booking'];
$order = Order_booking::make($current_user->id, $booking);
$order->create_new();
if ($order->id) {
   $menu_item = $_POST['menu_item'];
   $quantity = $_POST['quantity'];

   $order_items = array();

   while (($item = current($menu_item)) && ($qty = current($quantity))) {
      $order_item = Order_contents::make($order->id, $item, $qty);
      $order_item->save();
      echo $item;
      $order_items[] = $order_item;
      next($menu_item);
      next($quantity);
```

```
    }
//    $redirect_url = (isset($_POST['redirect_url'])) ? $_POST['redirect_url'] :
"../my_bookings.php?booking={$booking}";

redirect_to("http://localhost/mannhotel/my_booking.php?booking={$booking}")
;
}
?>

<pre>
    <?php
//    print_r($_POST);
//    print_r($menu_item);
//    print_r($order);
//    print_r($order_items);
    ?>
</pre>
```

Update Order

```
<?php

require_once '../includes/initialize.php';
$current_user = $session->get_user_object();


$content_id = (isset($_POST['content_id'])) ? $_POST['content_id'] : false;
$content_status = (isset($_POST['content_status'])) ? $_POST['content_status'] :
false;
$response["updated"] = false;

if ($content_id & $content_status) {
    $order_item = Order_contents::find_by_id($content_id);
    $order_item->status = $content_status;
    $order_item->save();
    $response['content'] = $order_item;
    $response["updated"] = true;
}

echo json_encode($response);
```

Layouts

Booking Form

```
<div class="modal about-modal fade" id="myModal" tabindex="-1"
role="dialog">
    <div class="modal-dialog" role="document">
```

```
      <div class="modal-content">
        <div class="modal-header">
          <button type="button" class="close" data-dismiss="modal" aria-
label="Close"><span aria-hidden="true">&times;</span></button>

          <h4 class="modal-title">Make Your Reservation Now</h4>
        </div>
        <div class="modal-body book-form">
          <form action="./logic/book_table.php" method="post">
            <?php if (!$current_user->id): ?>
              <div class="phone_email">
                <label>First Name : </label>
                <div class="form-text">
                  <span class="fa fa-user" aria-hidden="true"></span>
                  <input type="text" name="first_name" placeholder="First
Name" required="">
                </div>
              </div>
              <div class="phone_email">
                <label>Last Name : </label>
                <div class="form-text">
                  <span class="fa fa-user" aria-hidden="true"></span>
                  <input type="text" name="last_name" placeholder="Last
Name" required="">
                </div>
              </div>
              <div class="phone_email phone_email1">
                <label>Email : </label>
                <div class="form-text">
                  <span class="fa fa-envelope" aria-
hidden="true"></span>
                  <input type="text" name="email" placeholder="Email"
required="">
                </div>
              </div>
              <div class="phone_email">
                <label>Password : </label>
                <div class="form-text">
                  <span class="fa fa-user" aria-hidden="true"></span>
                  <input type="text" name="password"
placeholder="Password" required="">
                </div>
              </div>

              <div class="phone_email">
```

```
                    <label>Phone Number : </label>
                    <div class="form-text">
                        <span class="fa fa-phone" aria-hidden="true"></span>
                        <input type="text" name="phone_no" placeholder="Phone
no" required="">
                    </div>
                </div>
                <div class="phone_email phone_email1">
                    <label>Address : </label>
                    <div class="form-text">
                        <span class="fa fa-map-marker" aria-
hidden="true"></span>
                        <input type="text" name="address" placeholder="Your
Address" required="">
                    </div>
                </div>
            <?php endif; ?>
            <div class="clearfix"></div>
            <div class="agileits_reservation_grid">
                <div class="span1_of_1">
                    <label>Check In : </label>
                    <div class="book_date">
                        <span class="fa fa-calendar" aria-hidden="true"></span>
                        <input class="date" id="check_in" type="text"
name="check_in" required="">
                    </div>
                </div>
                <div class="span1_of_1">
                    <label>Check Out : </label>
                    <div class="book_date">
                        <span class="fa fa-calendar" aria-hidden="true"></span>
                        <input class="date" id="check_out" type="text"
name="check_out" required="">
                    </div>
                </div>

                <div class="clearfix"></div>

                <div class="span1_of_1">
                    <label>Type : </label>
                    <!-- start_section_room -->
                    <div class="section_room">
                        <span class="fa fa-users" aria-hidden="true"></span>
                        <select class="mandeep" name="type"
onchange="change_country(this.value)" class="frm-field required sect">
```

```
                        <?php
                        $options = Facility::find_types();
                        include './layouts/options_list.php';
                        ?>
                      </select>
                    </div>
                  </div>
                  <div class="span1_of_1">
                    <label>No.of People : </label>
                    <!-- start_section_room -->
                    <div class="section_room">
                      <span class="fa fa-users" aria-hidden="true"></span>
                      <select class="mandeep" name="capacity"
onchange="change_country(this.value)" class="frm-field required sect">
                        <option value="1">1 People</option>
                        <option value="2">2 People</option>
                        <option value="3">3 People</option>
                        <option value="4">4 People</option>
                        <option value="5">More</option>
                      </select>
                    </div>
                  </div>
                  <div class="clearfix"></div>
                </div>
                <input name="redirect_url" type="hidden" readonly value="<?php
echo $_SERVER["REQUEST_URI"]; ?>"/>
                <input type="submit" value="Book Now" />
              </form>
            </div>
            <!-- Calendar -->
            <link rel="stylesheet" href="css/jquery-ui.css" />
            <script src="js/jquery-ui.js"></script>
            <script>
                        $(jquery_calendar());
            </script>
            <!-- //Calendar -->
          </div>
        </div>
      </div>
      <!-- //modal -->
      <!-- FlexSlider -->
      <script defer src="js/jquery.flexslider.js"></script>
      <script type="text/javascript">
                        $(window).load(function () {
                          $('.flexslider').flexslider({
```

```
                    animation: "slide",
                    controlsContainer: $(".custom-controls-container"),
                    customDirectionNav: $(".custom-navigation a")
                });
            });
</script>
```

Flex Slider

```
<!-- banner-text -->
<div class="banner-text">
    <div class="flexslider">
        <ul class="slides">
            <li>
                <h2>Welcome to <?= strtoupper(SITE_TITLE); ?></h2>
                <p>
                    Are you on a hunt for perfect family vacation or a business trip
getaway?
                    We are the place of your choice
                </p>
                <a href="#" class="more" data-toggle="modal" data-
target="#myModal">BOOK NOW</a>
            </li>
            <li>
                <h3>Have a GREAT Time with OUR</h3>
                <p>
                    Our Hotel is a perfect holiday destination both for the families with
kids and for individuals. Our glorious mountainous landscapes will lighten your
stay through all the seasons!
                </p>
                <a href="#" class="more" data-toggle="modal" data-
target="#myModal">BOOK NOW</a>
            </li>
            <li>
                <h3>We will be so PROUD to </h3>
                <p>
                    Be our GUEST
                </p>
                <a href="#" class="more" data-toggle="modal" data-
target="#myModal">BOOK NOW</a>
            </li>
        </ul>
    </div>
    <!-- modal -->
    <?php include 'booking_form.php'; ?>
    <!-- //FlexSlider -->
```

```
</div>
<!-- //banner-text -->
```

Header

```
<!Doctype html>
<?php
require_once('includes/initialize.php');
global $session;
$current_user = $session->get_user_object();
?>
<html lang="en">
   <head>
      <title><?php echo (isset($title)) ? $title : SITE_TITLE; ?></title>
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <script type="application/x-javascript"> addEventListener("load",
function() { setTimeout(hideURLbar, 0); }, false); function hideURLbar(){
/*window.scrollTo(0,1);*/ } </script>
      <!-- Custom Theme files -->
      <link href="css/bootstrap.css" type="text/css" rel="stylesheet"
media="all">
      <link href="css/style.css" type="text/css" rel="stylesheet" media="all">
      <link rel="stylesheet" href="css/flexslider.css" type="text/css"
media="screen" />
      <link rel="stylesheet" href="css/swipebox.css">
      <link rel="stylesheet" href="css/select2.css">
      <link href="css/custom.css" type="text/css" rel="stylesheet" media="all">
      <!-- //Custom Theme files -->
      <!-- font-awesome icons -->
      <link href="css/font-awesome.css" rel="stylesheet">
      <!-- //font-awesome icons -->
      <!-- js -->
      <script src="js/jquery-2.2.3.min.js"></script>
      <script src="js/jquery.validate.min.js"></script>
      <!-- //js -->
      <!-- web-fonts -->
      <link
href='//fonts.googleapis.com/css?family=Cinzel+Decorative:400,700,900'
rel='stylesheet' type='text/css'>
      <link
href='//fonts.googleapis.com/css?family=Roboto:400,100,100italic,300,300ital
ic,400italic,500,500italic,700,700italic,900,900italic' rel='stylesheet'
type='text/css'>
      <!-- //web-fonts -->
      <!-- start-smooth-scrolling -->
```

```
    <script type="text/javascript" src="js/select2.min.js"></script>
    <script type="text/javascript" src="js/move-top.js"></script>
    <script type="text/javascript" src="js/easing.js"></script>
    <script type="text/javascript" src="js/custom.js"></script>
    <script type="text/javascript" src="js/jsrender.js"></script>
    <script type="text/javascript">
      jQuery(document).ready(function ($) {
        $(".scroll").click(function (event) {
          event.preventDefault();
          $('html,body').animate({scrollTop: $(this.hash).offset().top}, 1000);
        });
      });
    </script>
    <!-- //end-smooth-scrolling -->
  </head>
  <body>
    <?php if ($session->message()): ?>
      <div class="container-fluid" id="page_message">
        <h4 class="text-danger text-center"><?php echo $session-
>message(); ?></h4>
      </div>
    <?php endif; ?>

    <div class="banner simple" id="page_header">

      <?php if (isset($custom_header) && $custom_header): ?>
      <?php else: ?>
        <!-- banner -->
        <?php include 'navigation.php'; ?>
        <?php if (isset($nav_only) && !$nav_only): ?>
          <?php include 'flex_slider.php'; ?>
        <?php endif; ?>
      <?php endif; ?>
    </div>
```

Navigation

```
<div class="header"><!-- header -->
  <div class="container">
    <nav class="navbar navbar-default">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle" data-toggle="collapse"
data-target="#bs-example-navbar-collapse-1">
          <span class="sr-only">Toggle navigation</span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
```

```
                <span class="icon-bar"></span>
            </button>
            <h1><a  href="index.php"><?php echo SITE_TITLE; ?></a></h1>
        </div>
        <!-- navbar-header -->
        <div class="collapse navbar-collapse" id="bs-example-navbar-
collapse-1">
            <ul class="nav navbar-nav navbar-right">
                <?php
                if ((basename($_SERVER["REQUEST_URI"]) == "mannhotel") ||
                        (basename($_SERVER["REQUEST_URI"]) == "index.php")
                ):
                    ?>
                    <li><a href="#about" class="scroll">About</a></li>
                    <li><a href="#menu" class="scroll">Menu</a></li>
                    <li><a href="#team" class="scroll">Team</a></li>
                    <li><a href="#facility" class="scroll">Facilities</a></li>
                <?php endif; ?>

                <?php if (isset($current_user->id)): ?>
                    <li><a href="my_booking.php">My Bookings</a></li>
                    <?php if ($current_user->is_admin()): ?>
                        <li class="dropdown subnav_container">
                            <a class="dropdown-toggle" data-toggle="dropdown"
href="#">Administrations</a>
                            <ul class="dropdown-menu">
                                <li>
                                    <a href="list_tables.php">Admin Area</a>
                                </li>
                                <li>
                                    <a href="booking_updates.php">Booking
Orders</a>
                                </li>
                            </ul>
                        </li>
                    <?php endif; ?>
                    <li><a href="logout.php">Logout <?php echo $current_user-
>name(); ?></a></li>
                <?php else: ?>
                    <li><a href="login.php">Login</a></li>
                <?php endif; ?>
            </ul>
            <div class="clearfix"> </div>
        </div>
    </nav>
```

```
    </div>
</div>
<!-- //header -->
```

Footer

```
<footer id="page_footer">
    <!-- address -->
    <div id="contact" class="address">
        <div class="container">
            <div class="agile-title">
                <h3> Contact Us</h3>
            </div>
            <ul>
                <li><i class="fa fa-map-marker" aria-hidden="true"></i>Broome
St, Canada, NY 10002, New York</li>
                <li><i class="fa fa-phone" aria-hidden="true"> </i>  <a
href="tel:<?php echo DEVELOPER_TEL; ?>"><?php echo DEVELOPER_TEL;
?></a></li>
                <li><i class="fa fa-envelope" aria-hidden="true"></i><a
href="mailto:<?php echo DEVELOPER_MAIL; ?>"> <?php echo DEVELOPER_MAIL;
?></a></li>
            </ul>
        </div>
    </div>
    <!-- //address -->
    <!-- contact -->
    <a id="hotel_location"/>
    <div class="contact">
        <div class="col-md-6 contact-left">
            <iframe
src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d57537.6
41430789925!2d-
74.03215321337959!3d40.719122105634035!2m3!1f0!2f0!3f0!3m2!1i1024!2i
768!4f13.1!3m3!1m2!1s0x89c24fa5d33f083b%3A0xc80b8f06e177fe62!2sNew
+York%2C+NY%2C+USA!5e0!3m2!1sen!2sin!4v1456152197129"
allowfullscreen=""></iframe>
        </div>
        <div class="col-md-6 contact-right">
            <div class="wthree-contact-row">
                <h4>Get In Touch</h4>
                <form action="#" method="post">
                    <input type="text" name="name" placeholder="Name"
required="">
                    <input class="email" type="text" name="email"
placeholder="Email" required="">
```

```
                <textarea placeholder="Message" name="message"
required=""></textarea>
                <input type="submit" value="SUBMIT">
            </form>
          </div>
        </div>
        <div class="clearfix"> </div>
    </div>
    <!-- //contact -->
    <!-- footer -->
    <div class="footer">
        <div class="container">
            <div class="social-icon">
                <a href="#" class="social-button twitter"><i class="fa fa-
twitter"></i></a>
                <a href="#" class="social-button facebook"><i class="fa fa-
facebook"></i></a>
                <a href="#" class="social-button google"><i class="fa fa-google-
plus"></i></a>
                <a href="#" class="social-button dribbble"><i class="fa fa-
dribbble"></i></a>
                <a href="#" class="social-button skype"><i class="fa fa-
skype"></i></a>
            </div>
            <p>
                © 2017 <?php echo SITE_TITLE; ?>. All rights reserved
                |
                Developed by <a href="#"><?php echo DEVELOPER_NAME; ?></a>
            </p>
        </div>
    </div>
</footer>
<!-- //footer -->
<!-- swipe box js -->
<script src="js/jquery.swipebox.min.js"></script>
<script type="text/javascript">
    jQuery(function ($) {
        $(".swipebox").swipebox();
    });
</script>
<!-- //swipe box js -->
<!-- smooth-scrolling-of-move-up -->
<script type="text/javascript">
    $(document).ready(function () {
        /*
```

```
        var defaults = {
        containerID: 'toTop', // fading element id
        containerHoverID: 'toTopHover', // fading element hover id
        scrollSpeed: 1200,
        easingType: 'linear'
        };
        */

//        $().UItoTop({easingType: 'easeOutQuart'});

    });
</script>
<!-- //smooth-scrolling-of-move-up -->
<!-- Bootstrap core JavaScript
================================================== -->
<!-- Placed at the end of the document so the pages load faster -->
<script src="js/bootstrap.js"></script>
</body>
</html>
```

New Order

```
<div id="openOrderPanel">
    <script>
        function renderForm() {
            $("#menu_item").select2();
        }
        function quantity_tag(for_item)
        {
            var lable = "<lable>" + for_item + "</lable>";
            return lable + "<input min=\"1\" value=\"1\" name=\"quantity[]\"
class=\"form-control\" type=\"number\" required placeholder=\"" + for_item +
" Quantity\" />";
        }

        function handle_quantity() {
            var sendBack = "";
            $("#menu_item :selected").each(function () {
                sendBack += quantity_tag($(this).text().trim());
            });
            $("#quantity").html(sendBack);
            console.log(sendBack);
        }
```

```
  </script>

  <div id="newOrderForm_modal" class="modal fade" role="dialog">
    <div class="modal-dialog">
      <form id="newOrderForm" class="modal-content" method="post"
action="./logic/order.php" enctype="multipart/form-data">
        <h2 class="modal-header">
          <button type="button" class="close" data-
dismiss="modal">&times;</button>
          Order and proceed.
        </h2>
        <div class="modal-body">
          <label class="col-form-label" for="menu_item">Menu
Item</label>

          <label class="col-form-label" for="menu_item">Items in your
order</label>
          <select id="menu_item" name="menu_item[]" class="form-
control" multiple required onchange="handle_quantity();">
            <?php
            $options = Menu::find_all();
            include '../layouts/options_list.php';
            ?>
          </select>

          <label class="col-form-label" for="quantity">Quantity</label>
          <div id="quantity">
            Select a Menu Item to continue...
          </div>

<p>Once order is under process you can not cancel it.</p>
        </div>
        <input name="booking" type="hidden" readonly value="<?php echo
$current_booking->id; ?>"/>
        <input name="redirect_url" type="hidden" readonly value="<?php
echo $_SERVER["SERVER_NAME"] . "/mannhotel/my_booking.php"; ?>"/>
        <div class="modal-footer">
          <button class="btn btn-success" type="submit">
            <span class="glyphicon glyphicon-cutlery"></span>Order Now
          </button>
        </div>
      </form>
    </div>
  </div>
</div>
```

Options List

```
<?php
$selected_option = (isset($selected_option)) ? $selected_option : 0;
while ($option = current($options)):
   if ($option instanceof DatabaseObject) {
      ?>
      <option title="<?php echo trim($option->title()); ?>" value="<?php echo
$option->id; ?>"  <?php echo ($selected_option == $option->id) ? 'selected' :
''; ?> >
         <?php echo trim($option->name()); ?>
      </option>
   <?php } else {
      ?>
      <option value="<?php echo $option; ?>"  <?php echo ($selected_option
== $option) ? 'selected' : ''; ?> >
         <?php echo $option; ?>
      </option>
      <?php
   }
   ?>
   <?php
   next($options);
endwhile;
?>
```

Table Renderer

```
<table class="table table-striped ">

   <?php

   $edit_records = (isset($edit_records)) ? $edit_records : TRUE;
   $renderRow = false;
   if (method_exists(current($table_records), "renderTableHeader")) {
      $renderRow = true;
      echo current($table_records)->renderTableHeader();
   } else {
      ?>
      <thead>
         <tr class="row">
            <?php
            foreach ($table_records[0]->get_db_fields() as $key):
               ?>
               <th class="col-sm-12 col-md-2 ">
```

```
                <?php
                echo ucwords($key);
            endforeach;
            ?>
        </th>
    </tr>
</thead>
<tbody>
    <?php
}
while ($row = current($table_records)) :
    ?>
    <?php
    if (method_exists($row, "renderTableRow") && $renderRow) {
        echo $row->renderTableRow($edit_records);
    } else {
        // Render generic version
        ?>
        <tr class="row">
            <?php foreach ($row as $value): ?>
                <?php ?>
                <td class="col-sm-12 col-md-2">
                    <?php echo $value; ?>
                </td>

            <?php endforeach; ?>
        </tr>
        <?php
    }
    next($table_records);
endwhile;
?>
</tbody>
</table>
```

Admin Forms

Account

```
<?php
$tbClass = ucfirst($table);
if (isset($_GET['id'])) {
    $object = $tbClass::find_by_id($_GET['id']);
    $table_title = "Edit " . $object->name();
} else {
    $object = new $tbClass();
    $table_title = "Insert new Record in {$table}";
```

```
}
$object->init_members();
?>

<div class="container-fluid">

    <div class="panel panel-default col-md-8 col-md-offset-2">
        <h3 class="panel-heading text-capitalize"><?php echo $table_title;
?></h3>

        <form id="form" class="panel-body" method="post"
action="tableForms/insert.php" enctype="multipart/form-data">
            <?php if ($object->id): ?>
                <div class="form-group col-md-2">
                    <label class="col-form-label" for="id">Id</label>
                    <input id="id" name="id" class="form-control" type="number"
readonly value="<?php echo $object->id; ?>"/>
                </div>
            <?php endif; ?>
            <input id="redirect_url" name="redirect_url" type="hidden" readonly
value="<?php echo $_SERVER["REQUEST_URI"]; ?>"/>

            <div class="form-group col-md-3">
                <label class="col-form-label" for="user_name">User Name</label>
                <input id="user_name" name="user_name" class="form-control"
type="text" required value="<?php echo $object->user_name; ?>"/>
            </div>

            <div class="form-group col-md-3">
                <label class="col-form-label" for="authority_level">Authority
Level</label>
                <select id="authority_level" name="authority_level" class="form-
control">
                    <option value="1">1</option>
                    <option value="2">2</option>
                    <option value="3">3</option>
                </select>

            </div>

            <div class="form-group col-md-3">
                <label class="col-form-label" for="person">Person</label>

                <select id="person" name="person" class="form-control" required>
                    <option value="0">No One</option>
```

```
                <?php
                $selected_option = $object->person;
                $options = Person::find_all();
                include './layouts/options_list.php';
                ?>
              </select>
            </div>



            <div class="form-group col-md-4">
              <label class="col-form-label" for="password">Password</label>
              <input id="password" name="password" class="form-control"
type="password" required/>
            </div>
            <div class="form-group col-md-4">
              <label class="col-form-label" for="password2">Repeat
Password</label>
              <input id="password2" name="password2" class="form-control"
type="password" required/>
            </div>


            <div class="row btn-group-vertical col-md-6 col-md-offset-3">
              <input id="table_name" name="table_name" type="hidden"
value="<?php echo $table; ?>"/>
              <input class="form-control btn  btn-primary" type="submit"
value="Submit"/>
              <input class="form-control btn " type="reset" value="Clear"/>
            </div>
          </form>
        </div>
      </div>

      <script>
        var formRules = {
          rules: {
            user_name: {
              required: true,
              minlength: 6
            },
            password2: {
              required: true,
              minlength: 6,
              equalTo: "#password",
```

```
            maxlength: 30
          }

      },
      messages: {
      }
   };
</script>
```

Booking Form

```
<?php
$tbClass = ucfirst($table);
if (isset($_GET['id'])) {
   $object = $tbClass::find_by_id($_GET['id']);
   $table_title = "Edit " . $object->name();
} else {
   $object = new $tbClass();
   $table_title = "Insert new Record in {$table}";
}
$object->init_members();
?>

<div class="container-fluid">

   <div class="panel panel-default col-md-8 col-md-offset-2">
      <h3 class="panel-heading text-capitalize"><?php echo $table_title;
?></h3>

      <form id="form" class="panel-body" method="post"
action="tableForms/insert.php" enctype="multipart/form-data">
         <?php if ($object->id): ?>
            <div class="form-group col-md-2">
               <label class="col-form-label" for="id">Id</label>
               <input id="id" name="id" class="form-control" type="number"
readonly value="<?php echo $object->id; ?>"/>
            </div>
         <?php endif; ?>
         <input id="redirect_url" name="redirect_url" type="hidden" readonly
value="<?php echo $_SERVER["REQUEST_URI"]; ?>"/>
         <div class="form-group col-md-4">
            <label class="col-form-label" for="account">Account</label>
            <select id="account" name="account" class="form-control"
required>
               <?php
               $selected_option = $object->account;
```

```
                $options = Account::find_all();
                include './layouts/options_list.php';
                ?>
              </select>
            </div>

            <div class="form-group col-md-5">
              <label class="col-form-label" for="facility">Facility</label>
              <select id="facility" name="facility" class="form-control" required>
                <?php
                $options = Facility::find_all();
                $selected_option = $object->facility;
                include './layouts/options_list.php';
                ?>
              </select>
            </div>
            <div class="form-group col-md-4">
              <label class="col-form-label" for="check_in">Check In</label>
              <input id="check_in" name="check_in" class="form-control"
type="datetime-local" required value="<?php echo
DatabaseObject::form_date($object->check_in); ?>"/>
            </div>
            <div class="form-group col-md-4">
              <label class="col-form-label" for="check_out">Check Out</label>
              <input id="check_out" name="check_out" class="form-control"
type="datetime-local" required value="<?php echo
DatabaseObject::form_date($object->check_out); ?>"/>
            </div>

            <div class="row btn-group-vertical col-md-6 col-md-offset-3">
              <input id="table_name" name="table_name" type="hidden"
value="<?php echo $table; ?>"/>
              <input class="form-control btn  btn-primary" type="submit"
value="Submit"/>
              <input class="form-control btn " type="reset" value="Clear"/>
            </div>
          </form>
        </div>
      </div>

      <script>
        var formRules = {
          rules: {
          },
          messages: {
```

```
      }
   };
</script>
```

Discount Coupons Form

```
<?php
$tbClass = ucfirst($table);
if (isset($_GET['id'])) {
   $object = $tbClass::find_by_id($_GET['id']);
   $table_title = "Edit " . $object->name();
} else {
   $object = new $tbClass();
   $table_title = "Insert new Record in {$table}";
}
$object->init_members();
?>

<div class="container-fluid">

   <div class="panel panel-default col-md-8 col-md-offset-2">
      <h3 class="panel-heading text-capitalize"><?php echo $table_title;
?></h3>

      <form id="form" class="panel-body" method="post"
action="tableForms/insert.php" enctype="multipart/form-data">
         <?php if ($object->id): ?>
            <div class="form-group col-md-2">
               <label class="col-form-label" for="id">Id</label>
               <input id="id" name="id" class="form-control" type="number"
readonly value="<?php echo $object->id; ?>"/>
            </div>
         <?php endif; ?>

         <div class="form-group col-md-4">
            <label class="col-form-label" for="coupon">Coupon</label>
            <input id="coupon" name="coupon" class="form-control"
type="text" required value="<?php echo $object->coupon; ?>"/>
         </div>
         <div class="form-group col-md-6">
            <label class="col-form-label" for="img">Image</label>
            <input id="img" name="img" class="form-control" type="file"
accept="image/*" <?php if (!$object->id) echo 'required'; ?>/>
         </div>
```

```
        <input id="redirect_url" name="redirect_url" type="hidden" readonly
value="<?php echo $_SERVER["REQUEST_URI"]; ?>"/>
        <div class="form-group col-md-6">
          <label class="col-form-label" for="valid_through">Valid
Through</label>
          <input id="valid_through" name="valid_through" class="form-
control" type="date" required value="<?php echo $object->valid_through;
?>"/>
        </div>
        <div class="form-group col-md-4">
          <label class="col-form-label" for="valid_for">Valid For</label>
          <select id="valid_for" name="valid_for" class="form-control"
required>
            <option value="0">Everyone</option>
            <?php
            $selected_option = $object->valid_for;
            $options = Account::find_all();
            include './layouts/options_list.php';
            ?>
          </select>
        </div>

        <div class="form-group col-md-3">
          <label class="col-form-label" for="type">Type</label>
          <select class="form-control" name="type">
            <option value="special">Special</option>
            <option value="rate cutter">Rate Cutter</option>
            <option value="normal">Normal</option>
          </select>
        </div>
        <div class="form-group col-md-6">
          <label class="col-form-label" for="value">Value</label>
          <input id="value" name="value" class="form-control" type="number"
required value="<?php echo $object->value; ?>"/>
        </div>
        <div class="form-group col-md-3">
          <label class="col-form-label" for="status">Status</label>
          <select class="form-control" name="status">
            <option value="Available">Available</option>
            <option value="Redeemed">Redeemed</option>
            <option value="Expired">Expired</option>
          </select>
        </div>
```

```
        <div class="row btn-group-vertical col-md-6 col-md-offset-3">
            <input id="table_name" name="table_name" type="hidden"
value="<?php echo $table; ?>"/>
            <input class="form-control btn  btn-primary" type="submit"
value="Submit"/>
            <input class="form-control btn " type="reset" value="Clear"/>
        </div>
    </form>
  </div>
</div>

<script>
  var formRules = {
    rules: {
    },
    messages: {
    }
  };
</script>
```

Facility Form

```
<?php
$tbClass = ucfirst($table);
if (isset($_GET['id'])) {
  $object = $tbClass::find_by_id($_GET['id']);
  $table_title = "Edit " . $object->name();
} else {
  $object = new $tbClass();
  $table_title = "Insert new Record in {$table}";
}
$object->init_members();
?>

<div class="container-fluid">

  <div class="panel panel-default col-md-8 col-md-offset-2">
    <h3 class="panel-heading text-capitalize"><?php echo $table_title;
?></h3>

    <form id="form" class="panel-body" method="post"
action="tableForms/insert.php" enctype="multipart/form-data">
      <?php if ($object->id): ?>
        <div class="form-group col-md-2">
          <label class="col-form-label" for="id">Id</label>
```

```
            <input id="id" name="id" class="form-control" type="number"
readonly value="<?php echo $object->id; ?>"/>
          </div>
        <?php endif; ?>
        <input id="redirect_url" name="redirect_url" type="hidden" readonly
value="<?php echo $_SERVER["REQUEST_URI"]; ?>"/>
        <div class="form-group col-md-4">
          <label class="col-form-label" for="number">Title</label>
          <input id="title" name="title" class="form-control" type="text"
required value="<?php echo $object->title; ?>"/>
        </div>
        <div class="form-group col-md-4">
          <label class="col-form-label" for="type">Type</label>
          <select class="form-control" name="type">
            <?php
            $options = Facility::find_types();
            $selected_option = $object->type;
            include './layouts/options_list.php';
            ?>
          </select>
        </div>

        <div class="form-group col-md-2">
          <label class="col-form-label" for="number">Number</label>
          <input id="number" name="number" class="form-control"
type="number" required value="<?php echo $object->number; ?>"/>
        </div>
        <div class="form-group col-md-2">
          <label class="col-form-label" for="floor">Floor</label>
          <input id="floor" name="floor" class="form-control" type="number"
required value="<?php echo $object->floor; ?>"/>
        </div>
        <div class="form-group col-md-4">
          <label class="col-form-label" for="capacity">Capacity</label>
          <input id="capacity" name="capacity" class="form-control"
type="number" required value="<?php echo $object->capacity; ?>"/>
        </div>

        <div class="form-group col-md-4">
          <label class="col-form-label" for="charges">Charges</label>
          <input id="charges" name="charges" class="form-control"
type="number" required value="<?php echo $object->charges; ?>"/>
        </div>

        <div class="form-group col-md-4">
```

```
            <label class="col-form-label" for="available">Available</label>
            <input id="available" name="available" class="form-control"
type="number" required value="<?php echo $object->available; ?>"/>
        </div>
        <div class="form-group col-md-6">
            <label class="col-form-label" for="img">Image</label>
            <input id="img" name="img" class="form-control" type="file"
accept="image"<?php if (!$object->id) echo 'required'; ?>/>
        </div>

        <div class="row btn-group-vertical col-md-6 col-md-offset-3">
            <input id="table_name" name="table_name" type="hidden"
value="<?php echo $table; ?>"/>
            <input class="form-control btn  btn-primary" type="submit"
value="Submit"/>
            <input class="form-control btn " type="reset" value="Clear"/>
        </div>
    </form>
  </div>
</div>

<script>
  var formRules = {
    rules: {
    },
    messages: {
    }
  };
</script>
```

Hotel Form

```
<?php
$tbClass = ucfirst($table);
if (isset($_GET['id'])) {
  $object = $tbClass::find_by_id($_GET['id']);
  $table_title = "Edit " . $object->name();
} else {
  $object = new $tbClass();
  $table_title = "Insert new Record in {$table}";
}
$object->init_members();
?>

<div class="container-fluid">
```

```
    <div class="panel panel-default col-md-8 col-md-offset-2">
      <h3 class="panel-heading text-capitalize"><?php echo $table_title;
?></h3>

      <form id="form" class="panel-body" method="post"
action="tableForms/insert.php" enctype="multipart/form-data">
        <?php if ($object->id): ?>
          <div class="form-group col-md-2">
            <label class="col-form-label" for="id">Id</label>
            <input id="id" name="id" class="form-control" type="number"
readonly value="<?php echo $object->id; ?>"/>
          </div>
        <?php endif; ?>
        <input id="redirect_url" name="redirect_url" type="hidden" readonly
value="<?php echo $_SERVER["REQUEST_URI"]; ?>"/>
        <div class="form-group col-md-5">
          <label class="col-form-label" for="name">Name</label>
          <input id="name" name="name" class="form-control" type="text"
required value="<?php echo $object->name; ?>"/>
        </div>

        <div class="form-group col-md-5">
          <label class="col-form-label" for="manager">Manager</label>
          <input id="manager" name="manager" class="form-control"
type="number" required value="<?php echo $object->manager; ?>"/>
        </div>
        <div class="form-group col-md-6">
          <label class="col-form-label" for="address">Address</label>
          <input id="address" name="address" class="form-control"
type="text" required value="<?php echo $object->address; ?>"/>
        </div>
        <div class="form-group col-md-6">
          <label class="col-form-label" for="img">Image</label>
          <input id="img" name="img" class="form-control" type="file"
accept="image" <?php if (!$object->id) echo 'required'; ?> />
        </div>
        <div class="row btn-group-vertical col-md-6 col-md-offset-3">
          <input id="table_name" name="table_name" type="hidden"
value="<?php echo $table; ?>"/>
          <input class="form-control btn  btn-primary" type="submit"
value="Submit"/>
          <input class="form-control btn " type="reset" value="Clear"/>

        </div>
      </form>
```

```
    </div>
</div>

<script>
    var formRules = {
        rules: {
        },
        messages: {
        }
    };
</script>
```

Invoice Form

```
<?php
$tbClass = ucfirst($table);
if (isset($_GET['id'])) {
    $object = $tbClass::find_by_id($_GET['id']);
    $table_title = "Edit " . $object->name();
} else {
    $object = new $tbClass();
    $table_title = "Insert new Record in {$table}";
}
$object->init_members();
?>

<div class="container-fluid">

    <div class="panel panel-default col-md-8 col-md-offset-2">
        <h3 class="panel-heading text-capitalize"><?php echo $table_title;
?></h3>

        <form id="form" class="panel-body" method="post"
action="tableForms/insert.php" enctype="multipart/form-data">
            <?php if ($object->id): ?>
                <div class="form-group col-md-2">
                    <label class="col-form-label" for="id">Id</label>
                    <input id="id" name="id" class="form-control" type="number"
readonly value="<?php echo $object->id; ?>"/>
                </div>
            <?php endif; ?>
            <input id="redirect_url" name="redirect_url" type="hidden" readonly
value="<?php echo $_SERVER["REQUEST_URI"]; ?>"/>
            <div class="form-group col-md-4">
                <label class="col-form-label" for="booking">Booking</label>
```

```
            <select id="booking" name="booking" class="form-control" required
>
                <?php
                $selected_option = $object->booking;
                $options = Booking::find_all();
                include './layouts/options_list.php';
                ?>
            </select>
        </div>
        <div class="form-group col-md-4">
            <label class="col-form-label" for="generation_date">Generation
Date</label>
            <input id="generation_date" name="generation_date" class="form-
control" type="datetime-local" required value="<?php echo
DatabaseObject::form_date($object->generation_date); ?>"/>
        </div>
        <div class="form-group col-md-2">
            <label class="col-form-label" for="total_amount">Total
Amount</label>
            <input id="total_amount" name="total_amount" class="form-control"
type="number" required value="<?php echo $object->total_amount; ?>"/>
        </div>
        <div class="form-group col-md-3">
            <label class="col-form-label" for="discount">Discount</label>
            <select id="discount" name="discount" class="form-control"
required>
                <?php
                $selected_option = $object->discount;
                $options = Discount_coupons::find_all();
                include './layouts/options_list.php';
                ?>
            </select>
        </div>
        <div class="form-group col-md-3">
            <label class="col-form-label" for="amount_payable">Amount
Payable</label>
            <input id="amount_payable" name="amount_payable" class="form-
control" type="number" required value="<?php echo $object-
>amount_payable; ?>"/>
        </div>
        <div class="form-group col-md-6">
            <label class="col-form-label" for="status">Status</label>
            <select class="form-control" name="status">
                <?php
                $selected_option = $object->status;
```

```
                $options = array("Generated","Paid","Over due","Waived");
                include './layouts/options_list.php';
                ?>
            </select>
        </div>


        <div class="row btn-group-vertical col-md-6 col-md-offset-3">
            <input id="table_name" name="table_name" type="hidden"
value="<?php echo $table; ?>"/>
            <input class="form-control btn  btn-primary" type="submit"
value="Submit"/>
            <input class="form-control btn " type="reset" value="Clear"/>
        </div>
    </form>
  </div>
</div>

<script>
  var formRules = {
    rules: {
    },
    messages: {
    }
  };
</script>
```

Menu Form

```
<?php
$tbClass = ucfirst($table);
if (isset($_GET['id'])) {
  $object = $tbClass::find_by_id($_GET['id']);
  $table_title = "Edit " . $object->name();
} else {
  $object = new $tbClass();
  $table_title = "Insert new Record in {$table}";
}
$object->init_members();
?>

<div class="container-fluid">

  <div class="panel panel-default col-md-8 col-md-offset-2">
    <h3 class="panel-heading text-capitalize"><?php echo $table_title;
?></h3>
```

```
      <form id="form" class="panel-body" method="post"
action="tableForms/insert.php" enctype="multipart/form-data">
         <?php if ($object->id): ?>
            <div class="form-group col-md-2">
               <label class="col-form-label" for="id">Id</label>
               <input id="id" name="id" class="form-control" type="number"
readonly value="<?php echo $object->id; ?>"/>
            </div>
         <?php endif; ?>
         <input id="redirect_url" name="redirect_url" type="hidden" readonly
value="<?php echo $_SERVER["REQUEST_URI"]; ?>"/>
         <div class="form-group col-md-5">
            <label class="col-form-label" for="item">Item</label>
            <input id="item" name="item" class="form-control" type="text"
required value="<?php echo $object->item; ?>"/>
         </div>
         <div class="form-group col-md-5">
            <label class="col-form-label" for="price">Price</label>
            <input id="price" name="price" class="form-control" type="number"
required value="<?php echo $object->price; ?>"/>
         </div>
         <div class="form-group col-md-5">
            <label class="col-form-label" for="description">Description</label>
            <input id="description" name="description" class="form-control"
type="text" required value="<?php echo $object->description; ?>"/>
         </div>
         <div class="form-group col-md-5">
            <label class="col-form-label" for="img">Image</label>
            <input id="img" name="img" class="form-control" type="file"
accept="image" <?php if (!$object->id) echo 'required'; ?> />
         </div>

         <div class="row btn-group-vertical col-md-6 col-md-offset-3">
            <input id="table_name" name="table_name" type="hidden"
value="<?php echo $table; ?>"/>
            <input id="hotel" name="hotel" type="hidden" required value="1"/>
            <input class="form-control btn  btn-primary" type="submit"
value="Submit"/>
            <input class="form-control btn " type="reset" value="Clear"/>
         </div>
      </form>
   </div>
</div>
```

```
<script>
   var formRules = {
      rules: {
      },
      messages: {
      }
   };
</script>
```

Order Booking From

```
<?php
$tbClass = ucfirst($table);
if (isset($_GET['id'])) {
   $object = $tbClass::find_by_id($_GET['id']);
   $table_title = "Edit " . $object->name();
} else {
   $object = new $tbClass();
   $table_title = "Insert new Record in {$table}";
}
$object->init_members();
?>

<div class="container-fluid">

   <div class="panel panel-default col-md-8 col-md-offset-2">
      <h3 class="panel-heading text-capitalize"><?php echo $table_title;
?></h3>

      <form id="form" class="panel-body" method="post"
action="tableForms/insert.php" enctype="multipart/form-data">
         <?php if ($object->id): ?>
            <div class="form-group col-md-2">
               <label class="col-form-label" for="id">Id</label>
               <input id="id" name="id" class="form-control" type="number"
readonly value="<?php echo $object->id; ?>"/>
            </div>
         <?php endif; ?>
         <input id="redirect_url" name="redirect_url" type="hidden" readonly
value="<?php echo $_SERVER["REQUEST_URI"]; ?>"/>
         <div class="form-group col-md-6 col-md-offset-3">
            <label class="col-form-label" for="date">Date</label>
            <input id="date" name="date" class="form-control" type="datetime-
local" required value="<?php echo DatabaseObject::form_date($object->date);
?>"/>
         </div>
```

```
        <div class="form-group col-md-5">
          <label class="col-form-label" for="id">Account</label>
          <select id="account" name="account" class="form-control"
required>
            <?php
            $selected_option = $object->account;
            $options = Account::find_all();
            include './layouts/options_list.php';
            ?>
          </select>
        </div>
        <div class="form-group col-md-5">
          <label class="col-form-label" for="booking">Booking</label>
          <select id="booking" name="booking" class="form-control"
type="number" required>
            <?php
            $selected_option = $object->booking;
            $options = Booking::find_all();
            include './layouts/options_list.php';
            ?>
          </select>
        </div>

        <div class="row btn-group-vertical col-md-6 col-md-offset-3">
          <input id="table_name" name="table_name" type="hidden"
value="<?php echo $table; ?>"/>
          <input class="form-control btn  btn-primary" type="submit"
value="Submit"/>
          <input class="form-control btn " type="reset" value="Clear"/>
        </div>
      </form>
    </div>
</div>

<script>
  var formRules = {
    rules: {
    },
    messages: {
    }
  };
</script>
```

Order Contents Form

```
<?php
```

```
$tbClass = ucfirst($table);
if (isset($_GET['id'])) {
   $object = $tbClass::find_by_id($_GET['id']);
   $table_title = "Edit " . $object->name();
} else {
   $object = new $tbClass();
   $table_title = "Insert new Record in {$table}";
}
$object->init_members();
?>

<div class="container-fluid">

   <div class="panel panel-default col-md-8 col-md-offset-2">
      <h3 class="panel-heading text-capitalize"><?php echo $table_title;
?></h3>

      <form id="form" class="panel-body" method="post"
action="tableForms/insert.php" enctype="multipart/form-data">
         <?php if ($object->id): ?>
            <div class="form-group col-md-2">
               <label class="col-form-label" for="id">Id</label>
               <input id="id" name="id" class="form-control" type="number"
readonly value="<?php echo $object->id; ?>"/>
            </div>
         <?php endif; ?>
         <input id="redirect_url" name="redirect_url" type="hidden" readonly
value="<?php echo $_SERVER["REQUEST_URI"]; ?>"/>
         <div class="form-group col-md-6">
            <label class="col-form-label" for="order_id">Order</label>
            <select id="order_id" name="order_id" class="form-control"
required>
               <?php
               $selected_option = $object->order_id;
               $options = Order_booking::find_all();
               include './layouts/options_list.php';
               ?>
            </select>
         </div>
         <div class="form-group col-md-4">
            <label class="col-form-label" for="menu_item">Menu Item</label>
            <select id="menu_item" name="menu_item" class="form-control"
required>
               <?php
               $selected_option = $object->menu_item;
```

```
                $options = Menu::find_all();
                include './layouts/options_list.php';
                ?>
            </select>
        </div>
        <div class="form-group col-md-2">
            <label class="col-form-label" for="quantity">Quantity</label>
            <input id="quantity" name="quantity" class="form-control"
type="number" required value="<?php echo $object->quantity; ?>"/>
        </div>
        <div class="form-group col-md-4">
            <label class="col-form-label" for="status">Status</label>

            <select class="form-control" name="status">
                <?php
                $selected_option = $object->status;
                $options = array("Booked","Under Process","On the
way","Delivered","Failed");
                include './layouts/options_list.php';
                ?>
            </select>
        </div>


        <div class="row btn-group-vertical col-md-6 col-md-offset-3">
            <input id="table_name" name="table_name" type="hidden"
value="<?php echo $table; ?>"/>
            <input class="form-control btn  btn-primary" type="submit"
value="Submit"/>
            <input class="form-control btn " type="reset" value="Clear"/>
        </div>
    </form>
  </div>
</div>

<script>
  var formRules = {
    rules: {
    },
    messages: {
    }
  };
</script>
```

Person Form

```
<?php
$tbClass = ucfirst($table);
if (isset($_GET['id'])) {
   $object = $tbClass::find_by_id($_GET['id']);
} else {
   $object = new $tbClass();
}
$object->init_members();
?>

<div class="container-fluid">

   <div class="panel panel-default col-md-8 col-md-offset-2">
      <h3 class="panel-heading text-capitalize">Insert new <?php echo $table;
?></h3>

      <form id="form" class="panel-body" method="post"
action="tableForms/insert.php" enctype="multipart/form-data">
         <?php if ($object->id): ?>
            <div class="form-group col-md-2">
               <label class="col-form-label" for="id">Id</label>
               <input id="id" name="id" class="form-control" type="number"
readonly value="<?php echo $object->id; ?>"/>
            </div>
         <?php endif; ?>
         <input id="redirect_url" name="redirect_url" type="hidden" readonly
value="<?php echo $_SERVER["REQUEST_URI"]; ?>"/>

         <div class="form-group col-md-5">
            <label class="col-form-label" for="first_name">First Name</label>
            <input id="first_name" name="first_name" class="form-control"
type="text" required value="<?php echo $object->first_name; ?>"/>
         </div>

         <div class="form-group col-md-5">
            <label class="col-form-label" for="last_name">Last Name</label>
            <input id="last_name" name="last_name" class="form-control"
type="text" required value="<?php echo $object->last_name; ?>"/>
         </div>

         <div class="form-group col-md-8">
            <label class="col-form-label" for="address">Address</label>
            <input id="address" name="address" class="form-control"
type="text" required value="<?php echo $object->address; ?>"/>
```

```
        </div>

        <div class="form-group col-md-4">
            <label class="col-form-label" for="email">Email</label>
            <input id="email" name="email" class="form-control" type="text"
required value="<?php echo $object->email; ?>"/>
        </div>
        <div class="form-group col-md-4">
            <label class="col-form-label" for="email">Phone No</label>
            <input id="phone" name="phone" class="form-control" type="text"
required value="<?php echo $object->phone; ?>"/>
        </div>

        <div class="form-group col-md-6">
            <label class="col-form-label" for="dob">Date of Birth</label>
            <input id="dob" name="dob" class="form-control" type="date"
required value="<?php echo $object->dob; ?>"/>
        </div>

        <div class="form-group col-md-6">
            <label class="col-form-label"
for="anniversary">Anniversary</label>
            <input id="anniversary" name="anniversary" class="form-control"
type="date" value="<?php echo $object->anniversary; ?>"/>
        </div>
        <div class="form-group col-md-12">
            <label class="col-form-label" for="img">Image</label>
            <input id="img" name="img" class="form-control" type="file"
accept="image/*" <?php if (!$object->id) echo 'required'; ?>/>
        </div>



        <div class="row btn-group-vertical col-md-6 col-md-offset-3">
            <input id="table_name" name="table_name" type="hidden"
value="<?php echo $table; ?>"/>
            <input class="form-control btn  btn-primary" type="submit"
value="Submit"/>
            <input class="form-control btn " type="reset" value="Clear"/>
        </div>
      </form>
   </div>
</div>

<script>
```

```
    var formRules = {
        rules: {
        },
        messages: {
        }
    };
</script>
```

Insertion into Table

```
<?php include '../includes/initialize.php'; ?>
<?php
$table = $_POST["table_name"];
if (isset($_POST['id']) && !empty($_POST['id'])) {
    $object = $table::find_by_id($_POST['id']);
} else {
    $object = new $table;
}
$attrs = $object->attributes();

foreach ($_POST as $attribute => $value) {
    if ($attribute == "table_name" || $attribute == "id") {
        continue;
    }
    if (array_key_exists($attribute, $attrs) && !empty($value)) {
        $object->$attribute = $value;
    } else if (property_exists($table, $attribute)) {
        $object->$attribute = $value;
    }
}
?>
<?php
if ($object->validate_attributes($object->insertion_attributes())) {
    if (!empty($_FILES['img']['name'])) {
        $object->img = $object->upload_img($_FILES['img']);
    }
    $object->save();
    if ($table == 'user' && !($session->get_user_object() instanceof Account)) {
        $session->login($object);
    }

    $redirect_url = (isset($_POST['redirect_url'])) ? $_POST['redirect_url'] :
"../index.php";
    redirect_to($redirect_url);
} else {
    echo 'Errors';
```

```
}
?>
<pre>
   <?php print_r($_POST); ?>
</pre>
<pre>
   <?php print_r($object); ?>
</pre>
```

Delete Record

```
<?php include '../includes/initialize.php'; ?>
<?php

admins_only();

$table = $_POST["table_name"];
if (isset($_POST['id']) && !empty($_POST['id'])) {
   $object = $table::find_by_id($_POST['id']);
   $object->delete();
   $redirect_url = (isset($_POST['redirect_url'])) ? $_POST['redirect_url'] :
"../index.php";
   redirect_to($redirect_url);
//    echo $redirect_url;
//    print_r($object);
}else{
   print_r($_POST);
}
```

Styling Information

```
body{
   margin:0;
   font-family: 'Roboto', sans-serif;
   background: #fff;
}
body a{
   -webkit-transition: 0.5s all;
   -moz-transition: 0.5s all;
   -o-transition: 0.5s all;
   -ms-transition: 0.5s all;
   transition: 0.5s all;
   text-decoration:none;
}
h1,h2,h3,h4,h5,h6{
   margin:0;
   font-family: 'Cinzel Decorative', cursive;
```

```
}
p{
    margin:0;
}
ul,label{
    margin:0;
    padding:0;
}
body a:hover{
    text-decoration:none;
}
/*-- banner --*/
.banner {
    min-height: 780px;
    background: url(../hotel_pics/hotels-1.jpg)no-repeat center 0px;
    background-size: cover;
    position: relative;
}

.banner.simple{
    min-height: 0px;
    padding-bottom: 20px;
}
.header {
    position: relative;
    padding-top: 2em;
}
/*-- logo --*/
.navbar-header h1 {
    font-size: 2.5em;
    display: block;
}
.navbar-header h1 a {
    color: #FFFFFF;
    text-decoration: none;
    display: inline-block;
}
/*-- //logo --*/
/*-- top-nav --*/
.navbar-default {
    background: none;
    border: none;
    margin: 0;
    min-height: inherit;
}
```

```
ul.nav.navbar-nav.navbar-right {
   margin: 0.6em 0 0;
}
.navbar-default .navbar-nav > .active > a, .navbar-default .navbar-nav >
.active > a:hover, .navbar-default .navbar-nav > .active > a:focus {
   color: #99abd5;
   background: none;
}
.navbar-nav > li {
   margin: 0 1.2em;
}
.navbar-default .navbar-nav > li > a {
   color: #fff;
   font-size: 1.2em;
   padding: 0 5px;
   position: relative;
   font-weight: 300;
}
.navbar-default .navbar-nav > li > a:hover, .navbar-default .navbar-nav > li >
a:focus,.navbar-default .navbar-nav li a.active {
   color: #ff8c00;
}
.navbar-default .navbar-nav > li > a:before {
   content: '';
   position: absolute;
   top: 10px;
   background: #ffffff;
   height: 2px;
   width: 16px;
   right: -26px;
   -webkit-transition: .5s all;
   -moz-transition: .5s all;
   transition: .5s all;
   -webkit-transform: rotate(-64deg);
   -moz-transform: rotate(-64deg);
   -o-transform: rotate(-64deg);
   -ms-transform: rotate(-64deg);
   transform: rotate(-64deg);
}
.navbar-default .navbar-nav > li:nth-child(6) > a:before {
   width: 0;
}
/*-- //top-nav --*/
.banner-text {
   text-align: center;
```

```
    padding: 13em 0;
}
.banner-text h2,.banner-text h3 {
    color: #fff;
    font-size: 4em;
}
.banner-text p {
    width: 62%;
    color: #fff;
    margin: 1em auto;
    font-size: 1.1em;
    font-weight: 300;
    line-height: 2em;
    letter-spacing: 4px;
}
.banner-text h6 {
    font-size: 3em;
    color: #fff;
}
.banner-text h6 span {
    margin: 0 0.3em;
}
.banner-text a.more {
    font-size: 1em;
    color: #fff;
    font-weight: 300;
    padding: .6em 1.5em;
    border: 3px solid #fff;
    display: inline-block;
    margin-top: 1em;
    text-decoration:none;
}
.banner-text a.more:hover {
    border-color: #639;
    background: #639;
}
/*-- //banner --*/
/*-- welcome --*/
.welcome,.menu,.gallery{
    padding:5em 0;
}
.agile-title{
    text-align:center;
}
.agile-title h3 {
```

```
    font-size: 2.5em;
    text-align: center;
    color: #000;
    margin-bottom: 2em;
    border: 2px dashed #000;
    display: inline-block;
    padding: .4em 1em .2em;
}
.welcome-img {
    overflow: hidden;
    -webkit-box-shadow: 0px 0px 10px 0px #656060;
    -moz-box-shadow: 0px 0px 10px 0px #656060;
    box-shadow: 0px 0px 10px 0px #656060;
}
.welcome-left .img-responsive {
    width: 100%;
}
.welcome-left-grids {
    padding: 0;
    margin-top: 1em;
}
.welcome-left-grids:nth-child(2) {
    padding-right: 0.5em;
}
.welcome-left-grids:nth-child(3) {
    padding-left: 0.5em;
}
p {
    font-size: 1em;
    color: #555;
    line-height: 2em;
    font-weight: 400;
}
.welcome p {
    letter-spacing: 1px;
}
.open-hours-row {
    margin-top: 2em;
    background: #ffe4c4;
    border: 2px dashed #ff8c00;
}
.open-hours-left {
    padding: 3.85em 0;
    text-align: center;
    -webkit-transition: .5s all;
```

```
    -moz-transition: .5s all;
    transition: .5s all;
}
.open-hours-left:nth-child(1) {
    background: #ff8c00;
    padding: 2.8em 0;
}
.open-hours-left:nth-child(3) {
    border-left: 2px dashed #ff8c00;
    border-right: 2px dashed #ff8c00;
}
.open-hours-left h4 {
    font-size: 1.5em;
    color: #fff;
    line-height: 1.8em;
}
.open-hours-left h6 {
    font-size: 1em;
    font-family: 'Roboto', sans-serif;
    color: #555;
    -webkit-transition: .5s all;
    -moz-transition: .5s all;
    transition: .5s all;
}
.open-hours-left:hover h6{
    color: #fff;
}
.open-hours-left h5 {
    font-size: 1.3em;
    color: #000;
    margin-top: 0.5em;
    font-weight: 700;
}
.open-hours-left:hover {
    background: #ff8c00;
}
/*-- //welcome --*/
img.zoom-img{
    -webkit-transform: scale(1, 1);
    transform: scale(1, 1);
    -moz-transform: scale(1, 1);
    -ms-transform: scale(1, 1);
    -o-transform: scale(1, 1);
    transition-timing-function: ease-out;
    -webkit-transition-timing-function: ease-out;
```

```
    -moz-transition-timing-function: ease-out;
    -ms-transition-timing-function: ease-out;
    -o-transition-timing-function: ease-out;
    -webkit-transition-duration: .5s;
    -moz-transition-duration: .5s;
    -ms-transition-duration: .5s;
    -o-transition-duration: .5s;
}
img.zoom-img:hover{
    -webkit-transform: scale(1.2);
    transform: scale(1.2);
    -moz-transform: scale(1.2);
    -ms-transform: scale(1.2);
    -o-transform: scale(1.2);
    -webkit-transition-timing-function: ease-in-out;
    -webkit-transition-duration: 750ms;
    -moz-transition-timing-function: ease-in-out;
    -moz-transition-duration: 750ms;
    -ms-transition-timing-function: ease-in-out;
    -o-transition-timing-function: ease-in-out;
    -ms-transition-duration: 750ms;
    -o-transition-duration: 750ms;
    overflow: hidden;
}
/*-- slid --*/
.slid {
    background: url(../hotel_pics/hotels-1.jpg)no-repeat center 0px;
    background-size: cover;
    padding: 5em 0;
    text-align: center;
}
.slid h4 {
    font-size: 3em;
    color: #fff;
}
.slid h5 {
    font-size: 3.5em;
    color: #fff;
    font-weight: 600;
    margin: 0.5em 0;
    display: inline-block;
    background: #ff8c00;
    -webkit-border-radius: 50%;
    -moz-border-radius: 50%;
    border-radius: 50%;
```

```
    width: 145px;
    height: 145px;
    padding: 0.6em 0;
    -webkit-transition:.5s all;
    -moz-transition:.5s all;
    transition:.5s all;
}
.slid h5 span {
    font-family: 'Roboto', sans-serif;
    display: block;
    font-size: .45em;
    font-weight: 100;
    margin-top: 5px;
    letter-spacing: 2px;
}
.slid p {
    color: #fff;
    width: 65%;
    margin: 0 auto;
}
.slid:hover h5 {
    background: #639;
    -webkit-transform: rotatez(360deg);
    -moz-transform: rotatez(360deg);
    -o-transform: rotatez(360deg);
    -ms-transform: rotatez(360deg);
    transform: rotatez(360deg);
}
/*-- //slid --*/
/*-- menu --*/
.menu-left {
    position: relative;
}
.menu-left img {
    width: 100%;
}
.menu-right {
    padding: 2em;
    -webkit-transform: translate(500px);
    -moz-transform: translate(500px);
    -o-transform: translate(500px);
    -ms-transform: translate(500px);
    transform: translate(500px);
    -webkit-transition: .5s all;
    -moz-transition: .5s all;
```

```
    transition: .5s all;
    position: absolute;
    right: 0;
    width: 57%;
    top: 0;
    background-color: rgba(102, 51, 153, 0.86);
    height: 100%;
}
.menu-right h4 {
    color: #fff;
    font-size: 1.5em;
}
.menu-right h5 {
    font-size: 2.5em;
    color: #000;
    margin: 0.5em 0;
    font-weight: 700;
}
.menu-right p {
    color: #fff;
    font-weight: 300;
}
.accordion li:hover .menu-right{
    -webkit-transform: translate(0px);
    -moz-transform: translate(0px);
    -o-transform: translate(0px);
    -ms-transform: translate(0px);
    transform: translate(0px);
}

.accordion {
    width: 100%;
    overflow: hidden;
    list-style: none;
}
.accordion li{
    float:left;
    width:20%;
    overflow:hidden;
    height:250px;
    -moz-transition:width 0.2s ease-out;
    -webkit-transition:width 0.2s ease-out;
    -o-transition:width 0.2s ease-out;
    transition:width 0.2s ease-out;
    background:#ff8c00;
```

```
}
.accordion li:first-of-type{
    -moz-border-radius:10px 0 0 10px;
    -webkit-border-radius:10px 0 0 10px;
    -o-border-radius:10px 0 0 10px;
    border-radius:10px 0 0 10px;
}
.accordion li:last-of-type{
    -moz-border-radius:0 10px 10px 0;
    -webkit-border-radius:0 10px 10px 0;
    -o-border-radius:0 10px 10px 0;
    border-radius:0 10px 10px 0;
}
.accordion li:hover img {
    width: 100%;
}
.accordion:hover li{
    width:10%;
}
.accordion:hover li .menu-left img {
    width: 152%;
}
.accordion li:hover{
    width:60%;
}
.accordion:hover li:hover .menu-left img {
    width: 100%;
}
/*-------------------------------------*\
      VERTICAL
\*-------------------------------------*/
#vertical{
    width:940px;
    height:300px;
}
#vertical li{
    height:20%;
    width:100%;
    -moz-transition:height 0.2s ease-out;
    -webkit-transition:height 0.2s ease-out;
    -o-transition:height 0.2s ease-out;
    transition:height 0.2s ease-out;
}
#vertical li:first-of-type{
    -moz-border-radius:10px 10px 0 0;
```

```
    -webkit-border-radius:10px 10px 0 0;
    -o-border-radius:10px 10px 0 0;
    border-radius:10px 10px 0 0;
}
#vertical li:last-of-type{
    -moz-border-radius:0 0 10px 10px;
    -webkit-border-radius:0 0 10px 10px;
    -o-border-radius:0 0 10px 10px;
    border-radius:0 0 10px 10px;
}
#vertical:hover li{
    height:10%;
    width:100%;
}
#vertical li:hover{
    height:60%;
    width:100%;
}
/*-- //menu --*/
/*-- team --*/
.team{
    background: url(../images/bg1.jpg)no-repeat center 0px;
    background-size: cover;
    padding: 5em 0;
    text-align: center;
}
/* Seventh Section */

.w3ls-effect {
    display: block;
    height: 300px;
    overflow: hidden;
    position: relative;
    -webkit-box-shadow: 0 0 19px 4px rgba(0, 0, 0, 0.65);
    -moz-box-shadow: 0 0 19px 4px rgba(0, 0, 0, 0.65);
    box-shadow: 0 0 19px 4px rgba(0, 0, 0, 0.65);
    -webkit-transition: all 0.5s;
    -moz-transition: all 0.5s;
    -ms-transition: all 0.5s;
    -o-transition: all 0.5s;
    transition: all 0.5s;
    opacity: 1;
    filter: alpha(opacity=100);
}
.w3ls-effect img {
```

```
    height: 100%;
    width: 100%;
    -webkit-transition: all 0.5s;
    -moz-transition: all 0.5s;
    -ms-transition: all 0.5s;
    -o-transition: all 0.5s;
    transition: all 0.5s;
}
.w3ls-effect:before {
    content: '';
    background-color: rgba(0, 0, 0, 0);
    left: 0;
    top: 0;
    right: 0;
    bottom: 0;
    position: absolute;
    -webkit-transition: all 0.5s;
    -moz-transition: all 0.5s;
    -ms-transition: all 0.5s;
    -o-transition: all 0.5s;
    transition: all 0.5s;
}
.w3ls-effect:hover:before {
    background-color: rgba(0, 0, 0, 0.8);
}
.w3ls-effect:hover .view-caption {
    -moz-transform: translateY(-100%);
    -o-transform: translateY(-100%);
    -ms-transform: translateY(-100%);
    -webkit-transform: translateY(-100%);
    transform: translateY(-100%);
    border-radius: 0 0 0 0;
}
.w3ls-effect:hover .social-icon {
    bottom: 25%;
    -moz-transform: translateY(-80%);
    -o-transform: translateY(-80%);
    -ms-transform: translateY(-80%);
    -webkit-transform: translateY(-80%);
    transform: translateY(-80%);
}
.w3ls-effect:hover .social-icon a {
    -webkit-animation-name: translate-transition;
    -webkit-animation-duration: 0.8s;
    -webkit-animation-timing-function: linear;
```

```
    -webkit-animation-iteration-count: 1;
    animation-name: translate-transition;
    animation-duration: 0.8s;
    animation-timing-function: liner;
    animation-iteration-count: 1;
}
.w3ls-effect .view-caption {
    background-color: #FFF;
    -webkit-transition: all cubic-bezier(0.27, 0.89, 0.95, 0.59) 0.5s;
    -moz-transition: all cubic-bezier(0.27, 0.89, 0.95, 0.59) 0.5s;
    -ms-transition: all cubic-bezier(0.27, 0.89, 0.95, 0.59) 0.5s;
    -o-transition: all cubic-bezier(0.27, 0.89, 0.95, 0.59) 0.5s;
    transition: all cubic-bezier(0.27, 0.89, 0.95, 0.59) 0.5s;
    padding: 20px;
    height: 29%;
    text-align: left;
    position: relative;
    z-index: 99;
    -webkit-border-radius: 25px 0 0 0;
    -moz-border-radius: 25px 0 0 0;
    border-radius: 25px 0 0 0;
}
.w3ls-effect .view-caption h4 {
    color: #5f1ca2;
    font-size: 1.3em;
}
.w3ls-effect .view-caption p{
    margin-top: 0.3em;
}
.w3ls-effect .social-icon {
    bottom: 0;
    padding: 10px;
    left: 0;
    position: absolute;
    right: 0;
    text-align: left;
    overflow: hidden;
    z-index: 9;
    -webkit-transition: all 0.5s cubic-bezier(0.27, 0.89, 0.95, 0.59);
    -moz-transition: all 0.5s cubic-bezier(0.27, 0.89, 0.95, 0.59);
    -ms-transition: all 0.5s cubic-bezier(0.27, 0.89, 0.95, 0.59);
    -o-transition: all 0.5s cubic-bezier(0.27, 0.89, 0.95, 0.59);
    transition: all 0.5s cubic-bezier(0.27, 0.89, 0.95, 0.59);
    -moz-transform: translateY(1000%);
    -o-transform: translateY(1000%);
```

```
    -ms-transform: translateY(1000%);
    -webkit-transform: translateY(1000%);
    transform: translateY(1000%);
}
.social-button {
    display: inline-block;
    background-color: #fff;
    width: 32px;
    height: 32px;
    margin: 0 3px;
    line-height: 32px;
    text-align: center;
    position: relative;
    overflow: hidden;
    opacity: .99;
    -webkit-border-radius: 20%;
    -moz-border-radius: 20%;
    border-radius: 20%;
}
.social-button:before {
    content: '';
    background-color: #000;
    width: 120%;
    height: 120%;
    position: absolute;
    top: 102%;
    left: -118%;
    -webkit-transition: all 0.35s cubic-bezier(0.31, -0.105, 0.43, 1.59);
    -moz-transition: all 0.35s cubic-bezier(0.31, -0.105, 0.43, 1.59);
    transition: all 0.35s cubic-bezier(0.31, -0.105, 0.43, 1.59);
}
.social-button .fa {
    font-size: 18px;
    vertical-align: middle;
    -webkit-transform: scale(0.8);
    -moz-transform: scale(0.8);
    -o-transform: scale(0.8);
    -ms-transform: scale(0.8);
    transform: scale(0.8);
    -webkit-transition: all 0.35s cubic-bezier(0.31, -0.105, 0.43, 1.59);
    -moz-transition: all 0.35s cubic-bezier(0.31, -0.105, 0.43, 1.59);
    transition: all 0.35s cubic-bezier(0.31, -0.105, 0.43, 1.59);
}
.social-button.facebook:before {
    background-color: #3B5998;
```

```
}
.social-button.facebook .fa {
   color: #3B5998;
}
.social-button.twitter:before {
   background-color: #3CF;
}
.social-button.twitter .fa {
   color: #3CF;
}
.social-button.google:before {
   background-color: #DC4A38;
}
.social-button.google .fa {
   color: #DC4A38;
}
.social-button.dribbble:before {
   background-color: #F26798;
}
.social-button.dribbble .fa {
   color: #F26798;
}
.social-button.skype:before {
   background-color: #00AFF0;
}
.social-button.skype .fa {
   color: #00AFF0;
}
.social-button:focus:before, .social-button:hover:before {
   top: -10%;
   left: -10%;
}
.social-button:focus .fa, .social-button:hover .fa {
   color: #fff;
   -webkit-transform: scale(1);
   -o-transform: scale(1);
   -moz-transform: scale(1);
   -ms-transform: scale(1);
   transform: scale(1);
}
.team .agile-title h3 {
   color: #fff;
   border-color:#fff;
}
/*-- //team --*/
```

```
/*-- gallery --*/
.gallery-grids.gallery-grids-mdl {
    padding: 2em 1em;
}
.hover {
    width: 100%;
    height: 100%;
    float: left;
    overflow: hidden;
    position: relative;
    text-align: center;
    cursor: default;
}
.hover .overlay {
    width: 100%;
    height: 100%;
    position: absolute;
    overflow: hidden;
    top: 0;
    left: 0;
}
.hover img {
    display: block;
    position: relative;
}
.hover h4 {
    text-transform: uppercase;
    color: #fff;
    text-align: center;
    position: relative;
    font-size: 1.5em;
    padding: 10px;
    letter-spacing: 1px;
}

.hover .button.info {
    display: inline-block;
    text-decoration: none;
    padding: 7px 14px;
    color: #fff;
    border: none;
    background-color: transparent;
    font-size: 1.1em;
    outline:none;
    font-weight: 300;
```

```
}
.hover .button.info:hover {
   -webkit-box-shadow: 0 0 5px #fff;
   -moz-box-shadow: 0 0 5px #fff;
   box-shadow: 0 0 5px #fff;
}
.ehover14 img{
   -webkit-transition: all 0.35s;
   -moz-transition: all 0.35s;
   transition: all 0.35s;
}
.ehover14:hover img{
   -ms-filter: brightness(1.4);
   filter: brightness(1.4);
   -webkit-filter: brightness(1.4);
}
.ehover14 .overlay {
   -moz-opacity: 0;
   opacity: 0;
   width: 235px;
   height: 185px;
   left: 16%;
   top: 11%;
   border: 5px solid #fff;
   -webkit-transition: opacity 0.35s, transform 0.35s;
   -moz-transition: opacity 0.35s, transform 0.35s;
   transition: opacity 0.35s, transform 0.35s;
   padding: 3em 0;
   -webkit-transform: scale(0);
   -moz-transform: scale(0);
   transform: scale(0);
   -o-transform: scale(0);
   -ms-transform: scale(0);
}
.ehover14:hover .overlay {
   background-color: rgba(0, 0, 0, 0.71);
}
.ehover14 .button  {
   color:         #FFF;
   padding: 1em 0;
   opacity: 0;
   -webkit-transition: opacity 0.35s, -webkit-transform 0.35s;
   -moz-transition: opacity 0.35s, transform 0.35s;
   transition: opacity 0.35s, transform 0.35s;
   -webkit-transform: translate3d(-150%,-400%,0);
```

```
    -moz-transform: translate3d(-150%,-400%,0);
    -o-transform: translate3d(-150%,-400%,0);
    -ms-transform: translate3d(-150%,-400%,0);
    transform: translate3d(-150%,-400%,0);
}

.ehover14 h4 {
    background-color: transparent;
    color:          #FFF;
    padding:0.5em 0 1em;
    -moz-opacity: 0;
    opacity: 0;
    -webkit-transition: opacity 0.5s, -webkit-transform 0.5s;
    -moz-transition: opacity 0.5s, transform 0.5s;
    transition: opacity 0.5s, transform 0.5s;
    -webkit-transform: translate3d(-150%,-400%,0);
    -moz-transform: translate3d(-150%,-400%,0);
    -o-transform: translate3d(-150%,-400%,0);
    transform: translate3d(-150%,-400%,0);
    -ms-transform: translate3d(-150%,-400%,0);
}
.ehover14:hover .button, .ehover14:hover h4{
    -moz-opacity: 1;
    opacity: 1;
    -webkit-transform: translate3d(0,0,0);
    -moz-transform: translate3d(0,0,0);
    -o-transform: translate3d(0,0,0);
    -ms-transform: translate3d(0,0,0);
    transform: translate3d(0,0,0);
}
.ehover14:hover .overlay{
    -moz-opacity: 1;
    opacity: 1;
    -webkit-transform:scale(1);
    -moz-transform:scale(1);
    transform:scale(1);
    -o-transform:scale(1);
    -ms-transform:scale(1);
}
/*-- //gallery --*/
/*-- address --*/
.address {
    background: #ffead1;
    padding: 4em 0;
}
```

```
.address .agile-title h3 {
    font-size: 2.2em;
    margin-bottom: 1.5em;
}
.address ul{
    padding:0;
    margin:0;
}
.address ul li {
    list-style-type: none;
    font-size: 1.3em;
    color: #333;
    display: inline-block;
    width: 33%;
    text-align: center;
}
.address ul li i.fa {
    font-size: 2em;
    color: #639;
    margin-right: 0.3em;
    vertical-align: middle;
}
.address ul li a{
    color:#333;
}
.address ul li a:hover{
    color:#ff8c00;
}
/*-- //address --*/
/*-- contact --*/
.contact h4 {
    font-size: 2em;
    color: #fff;
    margin-bottom: 1em;
}
.contact-left{
    padding: 0;
}
.contact-right {
    background: url(../images/bg1.jpg)no-repeat center 0px;
    background-size: cover;
    padding: 0;
}
.wthree-contact-row {
    padding: 5em;
```

```
    background: rgba(0, 0, 0, 0.53);
}
.contact-left iframe {
    width: 100%;
    height: 29.4em;
    border:none;
}
.contact-right input[type="text"] {
    width: 48.5%;
    color: #fff;
    outline: none;
    font-size: 1em;
    font-weight: 300;
    padding: .6em 1em;
    margin-bottom: 1em;
    -webkit-appearance: none;
    background: none;
    border: 1px solid #ffffff;
    border-radius: 5px;
    -webkit-border-radius: 5px;
    -moz-border-radius: 5px;
    -o-border-radius: 5px;
    -ms-border-radius: 5px;
}
.contact-right input.email {
    margin-left: 1em;
}
.contact-right  textarea {
    resize: none;
    width: 100%;
    color: #fff;
    font-size: 1em;
    font-weight: 300;
    outline: none;
    padding: .8em 1em;
    border: none;
    min-height: 8em;
    -webkit-appearance: none;
    background: none;
    border: 1px solid #fff;
    border-radius: 5px;
    -webkit-border-radius: 5px;
    -moz-border-radius: 5px;
    -o-border-radius: 5px;
    -ms-border-radius: 5px;
```

```
}
.contact-right input[type="submit"] {
    border: none;
    outline: none;
    color: #fff;
    background: #663399;
    width: 31%;
    padding: 0.8em;
    font-size: 1em;
    margin: 0.5em 0 0;
    -webkit-appearance: none;
    border-radius: 5px;
    -webkit-border-radius: 5px;
    -moz-border-radius: 5px;
    -o-border-radius: 5px;
    -ms-border-radius: 5px;
    transition: 0.5s all;
    -webkit-transition: 0.5s all;
    -moz-transition: 0.5s all;
    -ms-transition: 0.5s all;
    -o-transition: 0.5s all;
}
.contact-right  input[type="submit"]:hover {
    background:#ff8c00;
}
::-webkit-input-placeholder {
    color:#fff !important;
}
:-moz-placeholder { /* Firefox 18- */
    color:#fff !important;
}
::-moz-placeholder {  /* Firefox 19+ */
    color:#fff !important;
}
:-ms-input-placeholder {
    color:#fff !important;
}
/*-- //contact --*/
/*-- footer --*/
.footer {
    padding: 3em 0;
    text-align: center;
    background: #000;
}
.footer p {
```

```
    color: #fff;
    margin-top: .5em;
}
.footer p a{
    color: #fff;
}
.footer p a:hover{
    color: #988AEE;
}
/*-- //footer --*/
/*-- modal --*/
.modal-open .modal {
    background: rgba(0, 0, 0, 0.48);
}
.modal-body {
    padding: 2em;
    text-align: left;
}
.modal-dialog {
    margin: 6em auto 0;
}
.modal-body iframe {
    border: none !important;
    width: 100%;
    min-height: 300px;
}
.about-modal .modal-header {
    border: none;
    min-height: 2.5em;
    padding: 2em 2em;
    background: #ffebd0;
}
.about-modal button.close {
    color: #639;
    opacity: .9;
    font-size: 2.5em;
    outline:none;
}
h4.modal-title {
    color: #639;
    font-size: 1.3em;
    font-weight: 600;
}
/*-- //modal --*/
/*-- reservation --*/
```

```
.book-form input[type="text"], select#country, select#country1, select.mandeep
{
   width: 100%;
   color: #999;
   font-size: 1em;
   padding: 10px  10px 10px  40px;
   outline: none;
   background: #fff;
   border: 1px solid #dedede;
}
.phone_email{
   float:left;
}
.phone_email.phone_email1{
   float:right;
}
.phone_email, .phone_email1 {
   width: 48.5%;
   margin-bottom: 1.5em;
}
.cuisine {
   margin-bottom: 2em;
}
.book-form .form-text,.book_date,.section_room {
   position: relative;
}
.book-form label {
   font-size: 1em;
   color: #555;
   font-weight: 400;
   margin-bottom: .5em;
}
.span1_of_1 {
   float: left;
   list-style: none;
   width: 31.5%;
}
.span1_of_1:nth-child(2) {
   margin: 0 1em;
}
.book-form span {
   position: absolute;
   color: #ff8c00;
   font-size: 1em;
   top: 1em;
```

```
    left: 1.2em;
}
.book-form input[type="submit"] {
    text-transform: capitalize;
    width: 100%;
    background: #639;
    color: #ffffff;
    padding: 10px 0;
    border: none;
    font-size: 1em;
    margin-top: 2em;
    outline: none;
    font-weight: 600;
    -webkit-transition:.5s all;
    -moz-transition:.5s all;
    transition:.5s all;
}
.book-form input[type="submit"]:hover{
    background: #ff8c00;
    color:#fff;
}
.book-form ::-webkit-input-placeholder {
    color:#999 !important;
}
.book-form :-moz-placeholder { /* Firefox 18- */
    color:#999 !important;
}
.book-form ::-moz-placeholder {  /* Firefox 19+ */
    color:#999 !important;
}
.book-form :-ms-input-placeholder {
    color:#999 !important;
}
/*-- //reservation --*/
/*-- slider-up-arrow --*/
#toTop {
    display: none;
    text-decoration: none;
    position: fixed;
    bottom: 3%;
    right: 3%;
    overflow: hidden;
    width: 32px;
    height: 32px;
    border: none;
```

```
    text-indent: 100%;
    background: url("../images/move-up.png") no-repeat 0px 0px;
}
#toTopHover {
    width: 32px;
    height: 32px;
    display: block;
    overflow: hidden;
    float: right;
    opacity: 0;
    -moz-opacity: 0;
    filter: alpha(opacity=0);
}
/*-- //slider-up-arrow --*/

.mandy-btn{
    border: 3px solid #fff;
    padding: .6em 1.5em;
    font-weight: 300;
    margin-top: 1em;

}

#mLocation{
    position: fixed;
    top: 50px;
    right: 50px;
    color: white;
}


.mandy-btn:hover {
    border-color: #639;
    background: #639;
}
ul.orders-list{
    padding-top: 24px;
    padding-right: 24px;
}
.panel.panel-primary {
    border: 1px solid darkgoldenrod;
}
.panel.panel-primary .panel-heading{
    background: darkgoldenrod;
    margin: 0px -15px;
```

```
    color: white;
    border-bottom: 1px solid #886308;
}
.panel.panel-primary .panel-footer{
    background: white;
}
.panel-footer .btn-group{
    margin-bottom: 20px;
}

.text{
    padding-left: 4px;
}
.text-primary{
    color: darkgoldenrod;
}
.text.small{
    font-weight: 400;
    font-size: .9em;
}
.text.simple{
    font-style: normal;
}
#openOrderPanel{
    padding: 8px;
}
.mar-8{
    margin-left: 8px;
}
.mar-16{
    margin-left: 16px;
}
.pad-8{
    padding-left: 8px;
}
.pad-16{
    padding-left: 16px;
}
.btn.btn-primary{
    background: goldenrod;
    color: white;
    border: 1px solid darkgoldenrod;
}
.btn.btn-primary:hover{
    background: gold;
```

```
    box-shadow: 0px 0px 2px 0px rgba(0,0,0,0.6);
}

#booking_details{
    padding-top: 30px;
    margin-right: 100px;
}

div.bill,a.bill{
    color: #886308;
    background: white;
    position: absolute;
    font-size: 1.3em;
    font-weight: 400;
    right: -50px;
    top: -10px;
    padding: 8px;
    z-index: 2;
    /*box-shadow: 9px 7px 25px -9px rgba(199,134,54,0.6);*/
    box-shadow: 11px 15px 34px -16px rgba(0,0,0,0.07);
    border: 1px solid rgba(0,0,0,0.2);
}
a.bill:hover{
    text-decoration: none;
}
a.bill:focus{
    text-decoration: none;
    box-shadow: 11px 15px 34px -16px rgba(0,0,0,0.07);
}
a.bill:active{
    text-decoration: none;
    background: #886308;
    color: white;
}


#newOrderForm_modal .modal-content{
    padding: 20px;
}

::-webkit-scrollbar {
    background: rgba(0,0,0,0.25);
    border-radius: 8px;
    width: 8px;
}
```

```
::-webkit-scrollbar:horizontal {
    height: 8px;
}
::-webkit-scrollbar-thumb {
    background: goldenrod;
    border-radius: 8px;
    /*-webkit-box-shadow: inset 0 0 2px goldenrod;*/
}
.list-group.bookings::-webkit-scrollbar {
    background: rgba(0,0,0,0.05);
    border-radius: 8px;
    width: 4px;
}
.list-group.bookings::-webkit-scrollbar-thumb {
    background: goldenrod;
    border-radius: 8px;
}


.list-group.bookings{
    max-height: 640px;
    padding-right: 40px;
    overflow: overlay;
    border-bottom: 1px solid darkgoldenrod;
}

.list-group-item.mandy h4{
    padding-left: 1.3em;
}

.list-group-item.mandy{
    padding: 1em;
    margin: 1.3em;
    margin-left: auto;
    margin-right: auto;
}
.list-group-item.mandy *{
    color: darkgoldenrod;
}

.list-group-item.mandy{
    background: white;
}

.list-group-item.mandy.active *{
```

```
    color: #fff;
}
.list-group-item.mandy.active{
    background: goldenrod;
    border: 2px solid darkgoldenrod;
}
a.selector_mk{
    cursor: pointer;
}

h4.heading{
    font-weight: 800;
}
h4.subheading{
    padding: 1em;
    font-size: 1.2em;
    font-weight: 600;
    font-style: italic;
}

.list-group-item.mandy:first-child{
    background: url("../images/g5.jpg");
    background-color: hsl(0,0%,50%);
    background-blend-mode: darken;
    background-repeat: no-repeat;
}
.list-group-item.mandy:first-child * {
    font-size: 1em;
    color: #fff;
    display: inline-block;
    text-decoration:none;
}


#to_record {
    float: right;
    background: #ffffff;
    padding: 16px;
    border: 1px solid;
    box-shadow: 0px 0px 8px 0px rgba(0, 0, 0, 0.15);
    border-radius: 8px;
}
#to_record:after {
    content: "";
    clear: both;
```

```
}


li.subnav_container ul{
   color: white;
   background: black;
   list-style: none;
   position: absolute;
   top: 2em;
   padding: 2px;
}

li.subnav_container ul li{
   padding: 8px;
}
li.subnav_container ul li:hover a{
   color: #ff8c00;
   background: black;
}

li.subnav_container ul a,li.subnav_container ul a:hover,li.subnav_container ul
a:active{
   color: white;
   padding: 4px;
}



/*-- responsive-design --*/
@media(max-width:1440px){
   .contact-right input[type="text"] {
      width: 48.3%;
   }
}
@media(max-width:1366px){
   .banner {
      min-height: 740px;
   }
}
@media(max-width:1280px){
   .banner {
      min-height: 710px;
   }
   .banner-text h2, .banner-text h3 {
      font-size: 3.6em;
```

```
        color: goldenrod;
    }
    .banner-text {
        padding: 12em 0;
    }
    .banner-text p {
        letter-spacing: 2px;
    }
    .slid {
        padding: 4em 0;
    }
    .slid h4 {
        font-size: 2.8em;
    }
    .slid h5 {
        font-size: 3em;
        width: 130px;
        height: 130px;
        padding: 0.75em 0;
    }
    .agile-title h3 {
        margin-bottom: 1.8em;
    }
}
@media(max-width:1080px){
    .banner-text p {
        width: 70%;
    }
    .banner {
        min-height: 680px;
    }
    .welcome, .menu, .gallery {
        padding: 4em 0;
    }
    .agile-title h3 {
        font-size: 2.3em;
        padding: .5em 1em .3em;
    }
    .banner-text h2, .banner-text h3 {
        font-size: 3.4em;
    }
    .welcome-right {
        padding: 0;
    }
    .welcome p {
```

```
    letter-spacing: 0px;
}
.open-hours-left h4 {
    font-size: 1.2em;
}
.open-hours-left:nth-child(1) {
    padding: 2.25em 0;
}
.open-hours-left h5 {
    font-size: 1.1em;
}
.open-hours-left {
    padding: 3em 0;
}
.accordion:hover li .menu-left img {
    width: 214%;
}
.w3ls-effect {
    height: 246px;
}
.w3ls-effect .view-caption {
    padding: 16px;
    height: 31%;
}
.gallery-grids {
    padding: 0 .5em;
}
.gallery-grids.gallery-grids-mdl {
    padding: 1em 0.5em;
}
.ehover14 .overlay {
    height: 155px;
    left: 11%;
    top: 11%;
    padding: 2em 0;
}
.ehover14 h4 {
    padding: 0.5em 0 0.8em;
}
.address ul li {
    font-size: 1.1em;
}
.address ul li i.fa {
    font-size: 1.5em;
    margin-right: 0.2em;
```

```
    }
    .address {
        padding: 3em 0;
    }
    .wthree-contact-row {
        padding: 3em;
    }
    .contact-right input[type="text"] {
        width: 48%;
    }
    .contact-left iframe {
        height: 25.3em;
    }
    .contact-right input[type="submit"] {
        padding: 0.7em;
    }
    .banner-text {
        padding: 11em 0;
    }
    .footer {
        padding: 2.5em 0;
    }
}
@media(max-width:1024px){
    .banner {
        min-height: 655px;
    }
    .open-hours-row {
        margin-top: 1em;
    }
    .slid p {
        width: 75%;
    }
    .contact-right input.email {
        margin-left: 0.5em;
    }
    .contact-right input[type="text"] {
        width: 48.7%;
    }
}
@media(max-width:991px){
    .navbar-header h1 {
        font-size: 2.2em;
    }
    div#bs-example-navbar-collapse-1 {
```

```
    padding: 0;
}
.navbar-default .navbar-nav > li > a {
    font-size: 1.1em;
    padding: 0 6px;
}
.navbar-nav > li:nth-child(6) {
    margin-right: 0;
}
.banner-text h2, .banner-text h3 {
    font-size: 3.2em;
    letter-spacing: 2px;
}
.banner-text p {
    width: 72%;
    letter-spacing: 1px;
}
.banner {
    min-height: 625px;
}
.banner-text {
    padding: 10em 0;
}
.agile-title h3 {
    font-size: 2.1em;
}
.welcome-left {
    padding: 0 8em;
}
.welcome-left-grids {
    float: left;
    width: 50%;
}
.welcome-right {
    margin-top: 2em;
}
.open-hours-left {
    float: left;
    width: 25%;
}
.open-hours-left:nth-child(1) {
    padding: 2.21em 2em;
}
.slid h4 {
    font-size: 2.6em;
```

```
}
.slid h5 {
    font-size: 2.8em;
    padding: 0.8em 0;
}
.slid p {
    width: 87%;
}
.slid {
    padding: 3.5em 0;
}
.agile-title h3 {
    margin-bottom: 1.5em;
}
.menu-left img {
    width: 136%;
}
.accordion:hover li .menu-left img {
    width: 241%;
}
.menu-right {
    width: 75%;
}
.team {
    padding: 4em 0;
}
.team-row {
    padding: 0 5.2em;
}
.team-grids {
    float: left;
}
.team-grids:nth-child(1), .team-grids:nth-child(2) {
    margin-bottom: 2em;
}
.w3ls-effect {
    height: 290px;
}
.gallery-grids {
    float: left;
    width: 33.33%;
}
.ehover14 .overlay {
    height: 124px;
    left: 8%;
```

```
        top: 10%;
        padding: 1.6em 0;
        width: 191px;
    }
    .hover h4 {
        font-size: 1.3em;
    }
    .ehover14 h4 {
        padding: 0.3em 0 0.5em;
    }
    .w3ls-effect .view-caption {
        height: 27%;
    }
    .address ul li:nth-child(1) {
        width: 43%;
    }
    .address ul li {
        width: 27%;
    }
    .contact-left iframe {
        height: 20em;
    }
    .wthree-contact-row {
        padding: 3.5em 5em;
    }
    .contact-right input[type="submit"] {
        width: 20%;
    }
}
@media(max-width:800px){
    .navbar-nav > li {
        margin: 0 0.8em 0 1.5em;
    }
    .navbar-header h1 {
        font-size: 2em;
    }
    .header {
        padding-top: 1.5em;
    }
    ul.nav.navbar-nav.navbar-right {
        margin: 0.5em 0 0;
    }
    .banner-text h2, .banner-text h3 {
        font-size: 3em;
        letter-spacing: 2px;
```

```
}
.banner-text p {
    width: 65%;
    font-size: 1em;
}
div#bs-example-navbar-collapse-1 {
    padding: 0 15px;
}
.banner {
    min-height: 590px;
}
.banner-text {
    padding: 9em 0;
}
.flex-control-nav {
    bottom: -48%;
}
.flex-direction-nav a {
    top: 27%;
}
.welcome-left {
    padding: 0 6em;
}
.open-hours-left:nth-child(1) {
    padding: 2.25em 2em;
}
.slid h4 {
    font-size: 2.3em;
}
.slid h5 {
    font-size: 2.5em;
    padding: 0.8em 0;
    width: 110px;
    height: 110px;
}
.slid {
    padding: 3em 0;
}
.address .agile-title h3 {
    font-size: 1.8em;
    margin-bottom: 1.3em;
    padding: .5em 0.8em .3em;
}
.contact-right input[type="text"] {
    width: 49.1%;
```

```
    }
}
@media(max-width:768px){
    .banner {
        min-height: 550px;
    }
    .banner-text {
        padding: 7em 0;
    }
    .contact-left iframe {
        height: 17em;
    }
    .agile-title h3 {
        font-size: 1.8em;
    }
    .slid h4 {
        font-size: 2.1em;
    }
}
@media(max-width:767px){
    .header {
        padding: 1.5em 1.5em 0;
    }
    ul.nav.navbar-nav.navbar-right {
        margin: 0;
    }
    .navbar-default .navbar-nav > li > a {
        text-align: center;
    }
    .navbar-default .navbar-collapse, .navbar-default .navbar-form {
        border: none;
        background: #000;
    }
    .navbar-default .navbar-toggle {
        border-color: #ff8c00;
        background: #ff8c00;
        margin: 0;
    }
    .navbar-default .navbar-toggle .icon-bar {
        background-color: #FFF;
    }
    .navbar-default .navbar-toggle:hover, .navbar-default .navbar-toggle:focus {
        background-color: #ff8c00;
    }
    .navbar-default .navbar-nav > li > a:before {
```

```
        width: 0;
    }
    .navbar-default .navbar-nav > li > a {
        padding: 0.8em;
    }
    .navbar-nav > li {
        margin: 0;
    }
    div#bs-example-navbar-collapse-1 {
        background: rgba(0, 0, 0, 0.83);
        margin: 0;
        padding: 0;
        position: absolute;
        width: 100%;
        z-index: 999;
        border: 1px solid #fff;
        margin-top: 0.5em;
    }
    .navbar-default .navbar-nav > li > a:before {
        width: 0;
        display:none;
    }
}
@media(max-width:736px){
    .banner-text h2, .banner-text h3 {
        font-size: 2.7em;
    }
    .banner-text a.more {
        padding: .6em 1.2em;
    }
    .banner-text a.more {
        margin-top: 0.5em;
    }
    .banner {
        min-height: 515px;
    }
    .banner-text p {
        width: 61%;
        font-size: 1em;
        letter-spacing: 0px;
    }
    .banner-text {
        padding: 6em 0;
    }
    .modal-open .modal {
```

```
    padding: 0 !important;
}
.modal-dialog {
    margin: 10% auto 0;
    width: 90%;
}
.about-modal .modal-header {
    padding: 1.5em 2em;
}
.welcome-left {
    padding: 0 5em;
}
.slid h4 {
    font-size: 2em;
}
.slid p {
    width: 95%;
}
.slid h5 {
    font-size: 2em;
    padding: 0.8em 0;
    width: 90px;
    height: 90px;
}
.welcome, .menu, .gallery {
    padding: 3.5em 0;
}
.accordion li {
    height: 230px;
}
.menu-right h5 {
    font-size: 2em;
}
.menu-right {
    width: 79%;
}
.w3ls-effect {
    height: 285px;
}
.team-row {
    padding: 0 3.2em;
}
.ehover14 .overlay {
    height: 109px;
    padding: 1.2em 0;
```

```
        width: 176px;
    }
    .hover .button.info {
        padding: 5px 11px;
        font-size: 1em;
    }
    .address ul li {
        width: 100%;
        margin-top: 1em;
    }
    .address ul li:nth-child(1) {
        width: 100%;
        margin: 0;
    }
    .contact-right input[type="text"] {
        width: 49%;
    }
    .wthree-contact-row {
        padding: 3em 5em;
    }
    .footer {
        padding: 2em 0;
    }
}
@media(max-width:667px){
    .accordion li {
        height: 215px;
    }
    .menu-right {
        width: 85%;
        padding: 2em 1.5em;
    }
    .menu-right h5 {
        margin: 0.3em 0;
    }
    .team {
        padding: 3.5em 0;
    }
    .team-row {
        padding: 0 1.2em;
    }
    .ehover14 .overlay {
        height: 109px;
        padding: 1.2em 0;
        width: 168px;
```

```
        left: 5%;
        top: 7%;
    }
    .address .agile-title h3 {
        font-size: 1.6em;
    }
    .contact-right input[type="text"] {
        width: 48.9%;
    }
    .contact-right input[type="submit"] {
        width: 24%;
    }
}
@media(max-width:640px){
    .slid {
        padding: 2.5em 0 4em;
    }
    .welcome, .menu, .gallery {
        padding: 3em 0;
    }
    .menu-right {
        padding: 2.5em 1.2em;
    }
    .team-grids {
        padding: 0 .5em;
        width: 50%;
    }
    .team-grids:nth-child(1), .team-grids:nth-child(2) {
        margin-bottom: 1em;
    }
    .team-row {
        padding: 0 4em;
    }
    .w3ls-effect {
        height: 260px;
    }
    .ehover14 .overlay {
        height: 105px;
        width: 166px;
    }
}
@media(max-width:600px){
    .header {
        padding: 1.5em 1em 0;
    }
```

```
.modal-body {
    padding: 1.5em;
}
.span1_of_1 {
    width: 31%;
}
.accordion li {
    height: 196px;
}
.menu-right {
    padding: 1.5em 1.2em;
}
.menu-right h4 {
    font-size: 1.2em;
}
.menu-right h5 {
    font-size: 1.5em;
}
.team {
    padding: 3em 0;
}
.team-row {
    padding: 0 2em;
}
.gallery-grids {
    width: 50%;
    padding: .5em;
}
.gallery-grids.gallery-grids-mdl {
    padding: .5em;
}
.ehover14 .overlay {
    height: 126px;
    width: 196px;
    left: 12%;
    top: 13%;
}
.ehover14 h4 {
    padding: 0.5em 0 0.8em;
}
.address ul li {
    font-size: 1em;
}
.address ul li i.fa {
    font-size: 1.3em;
```

```
        margin-right: 0.5em;
    }
    .address .agile-title h3 {
        font-size: 1.4em;
    }
    .address {
        padding: 2.5em 0;
    }
    .contact-left iframe {
        height: 13em;
    }
    .wthree-contact-row {
        padding: 3em 3em;
    }
    .contact-right input[type="submit"] {
        width: 31%;
    }
    .banner-text h2, .banner-text h3 {
        font-size: 2.5em;
    }
    .banner {
        min-height: 490px;
    }
    .banner-text {
        padding: 5.5em 0;
    }
}
@media(max-width:568px){
    .accordion li {
        height: 182px;
    }
    .menu-right p {
        font-size: .9em;
    }
    .menu-right h5 {
        font-size: 1.8em;
        display: inline-block;
        margin: 0 .5em 0.5em;
    }
    .menu-right h4 {
        display: inline-block;
    }
    .w3ls-effect {
        height: 245px;
    }
```

```
    .ehover14 .overlay {
        height: 120px;
        width: 185px;
    }
    .contact-right input[type="text"] {
        width: 48.8%;
    }
}
@media(max-width:480px){
    .welcome-left {
        padding: 0 2em;
    }
    .open-hours-left h4 {
        font-size: 1em;
    }
    .open-hours-left:nth-child(1) {
        padding: 2.1em 0;
    }
    .open-hours-left {
        padding: 2.5em 0;
    }
    .slid h4 {
        font-size: 1.8em;
    }
    .slid p {
        width: 100%;
    }
    .accordion li {
        height: 158px;
    }
    .menu-left img {
        width: 121%;
    }
    .menu-right h5 {
        margin: 0 .5em 0.1em;
    }
    .menu-right {
        padding: 1em 1em;
        width: 95%;
    }
    .team-row {
        padding: 0 1em;
    }
    .w3ls-effect {
        height: 207px;
```

```
}
.w3ls-effect .view-caption {
    height: 37%;
}
.w3ls-effect .view-caption h4 {
    font-size: 1.2em;
}
.agile-title h3 {
    font-size: 1.6em;
    margin-bottom: 1em;
}
.ehover14 .overlay {
    height: 99px;
    width: 154px;
    padding: 0.6em 0;
}
.banner-text {
    padding: 5em 0;
}
.wthree-contact-row {
    padding: 2em 2em;
}
.contact-right input[type="text"] {
    width: 48.6%;
}
.contact h4 {
    font-size: 1.8em;
}
.navbar-toggle {
    padding: 8px 9px;
}
.navbar-header h1 {
    font-size: 1.8em;
    margin-top: 5px;
}
.banner-text h2, .banner-text h3 {
    font-size: 2.2em;
}
h4.modal-title {
    font-size: 1.2em;
}
.about-modal .modal-header {
    padding: 1.2em 2em;
}
.about-modal button.close {
```

```
        font-size: 2em;
        line-height: 0.8em;
    }
    .span1_of_1 {
        width: 100%;
    }
    .span1_of_1:nth-child(2) {
        margin: 1em 0;
    }
    .phone_email, .phone_email1 {
        width: 100%;
        margin-bottom: 0.8em;
        float: none;
    }
    .book-form label {
        font-size: 0.9em;
    }
    .book-form input[type="text"], select#country, select#country1 {
        font-size: .9em;
        padding: 8px 8px 8px 40px;
    }
    .book-form span {
        top: 0.7em;
    }
    .book-form input[type="submit"] {
        margin-top: 1em;
    }
    .modal-dialog {
        margin: 6% auto;
        width: 85%;
    }
}
@media(max-width:414px){
    .header {
        padding: 1em 0.5em 0;
    }
    .navbar-toggle {
        padding: 7px 7px;
    }
    .navbar-header h1 {
        font-size: 1.9em;
        margin-top: 2px;
    }
    .banner {
        min-height: 417px;
```

```
}
.banner-text {
    padding: 4em 0;
}
.banner-text p {
    width: 68%;
    font-size: 0.9em;
}
.banner-text a.more {
    margin-top: 0;
    font-size: 0.9em;
}
.welcome, .menu, .gallery {
    padding: 2.5em 0;
}
.welcome-left {
    padding: 0 1em;
}
p {
    font-size: 0.9em;
}
.welcome-right {
    margin-top: 1em;
}
.open-hours-left {
    float: none;
    width: 100%;
    padding: 1em 0;
}
.open-hours-left:nth-child(1) {
    padding: 1em 0;
}
.open-hours-left:nth-child(3) {
    border:none;
    border-top: 2px dashed #ff8c00;
    border-bottom: 2px dashed #ff8c00;
}
.open-hours-left h4 {
    font-size: 1.2em;
}
.slid h4 {
    font-size: 1.8em;
}
.accordion li {
    float: none;
```

```
    width: 100%;
    height: 190px;
}
.accordion:hover li {
    width: 100%;
}
.accordion li:hover {
    width: 100%;
}
.accordion:hover li .menu-left img {
    width: 100%;
}
.menu-left img {
    width: 100%;
    margin-top: -8em;
}
.menu-right {
    padding: 2em 1.5em;
    width: 72%;
}
.menu-right p {
    margin-top: .5em;
}
.team-row {
    padding: 0;
}
.team-grids {
    padding: 0 2em;
    width: 100%;
    float: none;
}
.team-grids:nth-child(3){
    margin-bottom: 1em;
}
.w3ls-effect {
    height: 340px;
}
.w3ls-effect .view-caption {
    height: 25%;
}
.gallery-grids {
    width: 100%;
}
.ehover14 .overlay {
    height: 160px;
```

```
        width: 250px;
        padding: 3em 0;
    }
    .modal-dialog {
        width: 90%;
    }
    h4.modal-title {
        font-size: 1em;
    }
    .contact-right input[type="text"] {
        width: 100%;
    }
    .contact-right input.email {
        margin-left: 0;
    }
    .contact-right input[type="submit"] {
        width: 39%;
        padding: 0.6em;
    }
    .contact h4 {
        font-size: 1.6em;
    }
}
@media(max-width:384px){
    .welcome-left {
        padding: 0;
    }
    .slid h4 {
        font-size: 1.6em;
    }
    .slid {
        padding: 2em 0 3em;
    }
    .slid h5 {
        font-size: 1.5em;
        padding: 0.8em 0;
        width: 70px;
        height: 70px;
    }
    .menu-right {
        padding: 2em 1.5em;
        width: 74%;
    }

    .about-modal .modal-header {
```

```
        padding: 1.2em 1em;
    }
}
@media(max-width:375px){
    .menu-right {
        width: 75%;
    }
    .ehover14 h4 {
        padding: 0em 0 1em;
    }
}
@media(max-width:320px){
    .header {
        padding: 1em 0 0;
    }
    .navbar-default .navbar-nav > li > a {
        padding: 0.7em;
        font-size: 1em;
    }
    .navbar-header h1 {
        font-size: 1.7em;
        margin-top: 3px;
    }
    .banner-text h2, .banner-text h3 {
        font-size: 1.7em;
    }
    .banner-text a.more {
        padding: .5em 1em;
    }
    p {
        font-size: 0.87em;
    }
    .modal-body {
        padding: 1.2em;
    }
    .book-form input[type="submit"] {
        font-size: 0.9em;
    }
    .accordion li {
        height: 230px;
    }
    .menu-right h5 {
        margin: 0.2em 0;
        display: block;
    }
```

```
    .menu-right {
        width: 86%;
        padding: 2.5em 1.5em;
    }
    .team-grids {
        padding: 0 1em;
    }
    .w3ls-effect {
        height: 268px;
    }
    .ehover14 .overlay {
        height: 131px;
        width: 205px;
        padding: 2.1em 0;
        left: 10%;
        top: 11%;
    }
    .address ul li {
        font-size: 0.9em;
    }
    .wthree-contact-row {
        padding: 2em 1.5em;
    }
    .contact-right input[type="text"] {
        font-size: 0.9em;
    }
    .contact-right input[type="submit"] {
        width: 46%;
        padding: 0.7em;
        font-size: 0.9em;
    }
}
/*-- //responsive-design --*/
```

Client Side Scripting

```
function showPopup(url) {
    newwindow = window.open(url, "Checklist",
'height=480,width=320,top=50,left=200,location=0,resizable');
    if (window.focus) {
        newwindow.focus()
    }
}
```

```
var previous_booking_id;
function get_details(booking_id,order_more,link_checkout) {
   if(order_more === undefined){
      order_more = true;
   }
   if(link_checkout === undefined){
      link_checkout = true;
   }

   $("#booking-" + previous_booking_id).removeClass("active");
   $("#booking-" + booking_id).addClass("active");
   $.post("./logic/booking_details.php",
         {
            id: booking_id,
            order_more: order_more,
            link_checkout: link_checkout
         },
         function (response) {
            $("#booking_details").html(response);
         }
   );
   previous_booking_id = booking_id;
}

function jquery_calendar() {
   format = "yy-mm-dd";
   check_in = $("input#check_in");
   check_out = $("input#check_out");
   $("input#check_in").datepicker()
         .on("change", function () {
            check_out.datepicker("option", "minDate", check_in.val());
         });
   $("input#check_out").datepicker()
         .on("change", function () {
            check_in.datepicker("option", "maxDate", check_out.val());
         });
   check_in.datepicker("option", "dateFormat", format);
   check_in.datepicker("option","minDate",0);
   check_out.datepicker("option", "dateFormat", format);

   $("#ui-datepicker-div").hide();
}
```

Comments and Description of Coding Segments

Index

This is the page where users lands when they input URL of our portal in their browser. Users see all the details there are related to the hotel. Which includes:

- Services Available
- Booking for a service
- Menu Items
- Facilities
- Introduction with the webmins
- Map location of the hotel

Login

To gain access of secure data users must go through this page and sign in using correct credentials and once they do they may visit other pages and access the data in them.

My Booking

All the bookings made by a user can be found on this very page. Upon clicking one of the bookings user can view details regarding the same and order services for that booking.

Once done, user can press on checkout and pay the bill.

Checkout

Once booking's stay is over, user can proceed to checkout from the hotel by confirming his billed amount and applying discounts on it if any. Once they do an invoice is generated for their filing purposes.

View Bill

Invoice generated during the checkout can be seen here and printed as well. The page is optimised to support printers and print only the required contents.

Admin List Tables

Admins of the site are allowed to view/edit/delete any record that is in the database to facilitate the hotel management and futher help the guests.

Also this page is able to generate reports for all the tables.

Database

This class facilitates the process of query database tables securely and easily after creating a conncetion with information provided in **Config.php** file.

Also this class handles code to get the record rows as an array from the result set we get after executing a query.

Insertion/Updation of row(s) also results in giving back the id of rows affected and their count as well.

Database Object

This class helps in conversion of record array into Table object to use them properly in the code. Also functions like *save(), update(), delete()* helps in performing the tasks more effectively.

Table Class

Every table has its own unique class to facilitate their personal tasks. Such as finding all the records, finding orders placed in a specific booking, generating invoice for a booking, etc.

Session

User login, logout, message handling and redirection tasks can not be done easily if we do not use sessions and session management is crucial for better code execution else it interferes with the actual code. That is why a separate class has been created for managing tasks mentioned before.

Standardization of the coding

Efforts have been made to write the entire code in a standard and formalised manner. For that purpose techniques like Object-Oriented programming, functions and methods, CSS for styling information and jQuery for client side experience are used.

PHP has the following conventions as well.

Naming Conventions

Global Variables and Functions

If your package needs to define global variables, their names should start with a single underscore followed by the package name and another underscore. For example, the PEAR package uses a global variable called $_PEAR_destructor_object_list.

Global functions should be named using the "studly caps" style (also referred to as "bumpy case" or "camel caps"). In addition, they should have the package name as a prefix, to avoid name collisions between packages. The initial letter of the name (after the prefix) is lowercase, and each letter that starts a new "word" is capitalized. An example:

XML_RPC_serializeData()

Classes

Classes should be given descriptive names. Avoid using abbreviations where possible. Class names should always begin with an uppercase letter. The PEAR class hierarchy is also reflected in the class name, each level of the hierarchy separated with a single underscore. Examples of good class names are:

Log	Net_Finger	HTML_Upload_Error

Class Variables and Methods

Class variables (a.k.a properties) and methods should be named using the "studly caps" style (also referred to as "bumpy case" or "camel caps"). Some examples (these would be "public" members):

$counter	connect()	getData()	buildSomeWidget()

Private class members are preceded by a single underscore. For example:

$_status	_sort()	_initTree()

Constants

Constants should always be all-uppercase, with underscores to separate words. Prefix constant names with the uppercased name of the class/package they are used in. Some examples:

DB_DATASOURCENAME	SERVICES_AMAZON_S3_LICENSEKEY

Code Efficiency

PHP's object orientation has been put to a good use in this project to make the code more and more efficient and light on the database as well.

Also using less as the css pre-processor has made the generation of stylesheet a lot easier.

jQuery helps in writing easy to read and tested code for client sides.

Error handling

PHP does a good job in handling the error prone code by itself and to ensure that the whole website is error free all the cases have been put inside if else blocks to test the values before performing any action.

Some examples are:

- Checking if Constant is already defined before creating it by calling defined function in php
- Using issset() to ensure that POST values from the Form were actually submitted
- Checking for null values before inserting them into database.
- Santizing the attributes and SQL calls to prevent faulty data.

Parameter Passing and Calling

Our website use two technologies PHP and JavaScript to create communication between a user and our system and following techniques are used to pass parameters in the website.

- **Client to Server:** Using Request parameters ie GET and POST.
    - This is sent using either a Form or JavaScript AJAX calls like $.ajax() and $.get(), $.post()
- **Server to Server:** If we are in same script this is done by function calls like find_by_id($id) and once we change scripts we need to send GET or POST request just like a client would.

By sending data by passing parameters we help in making our code more modular and easy to reuse as well.

Validation checks

HTML5 validates the data automatically if we instruct it in the form field using the tags like required, min, max, type.

But to make sure everything is tested before submission. We have used the jQuery validate data before submissions.

Screen Shots

Home Page

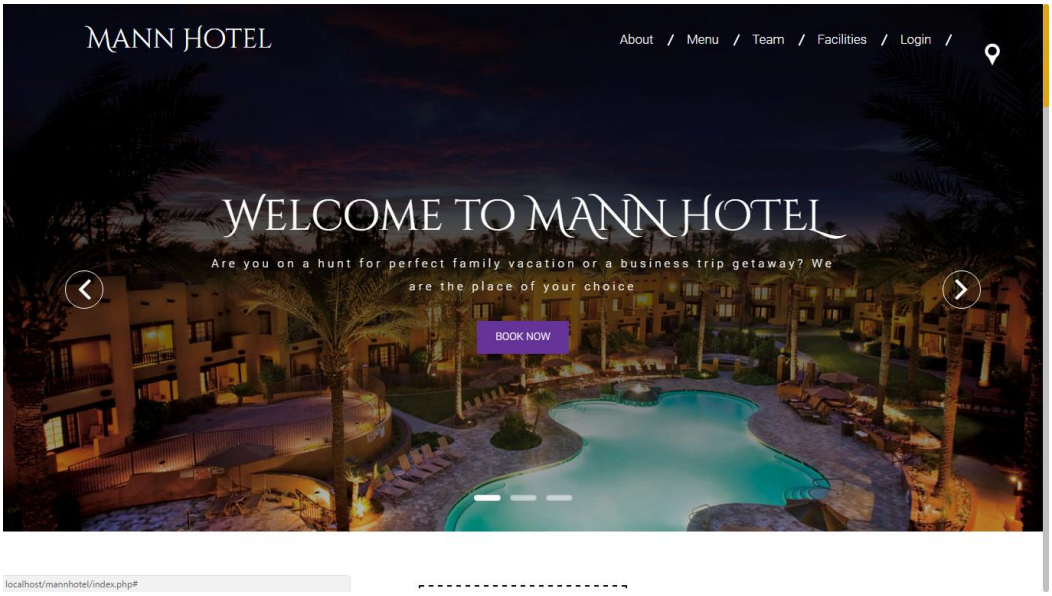


Figure 26 Site Home Page

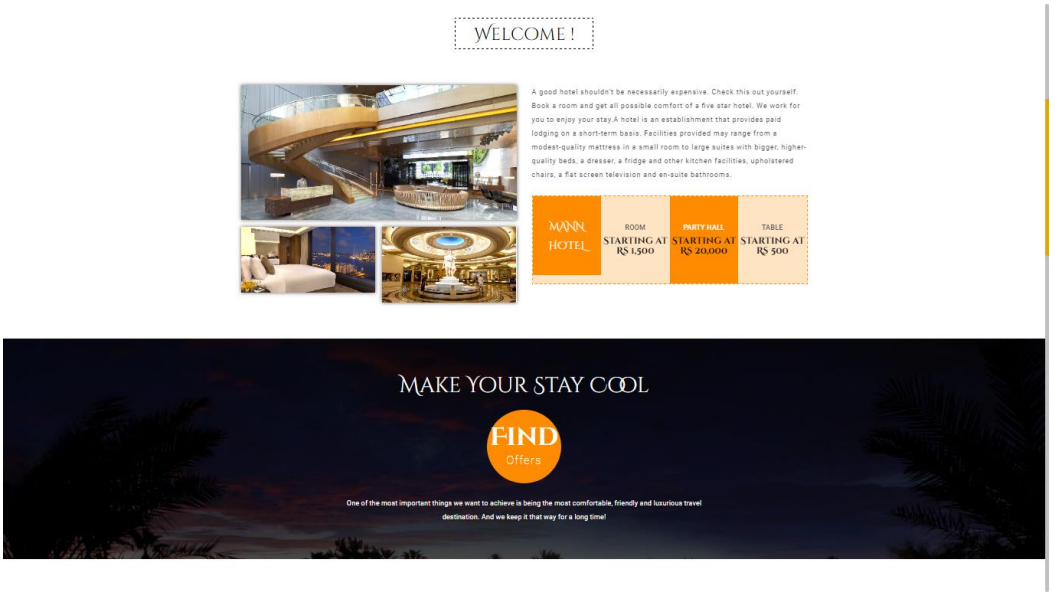


Figure 27 SIte View

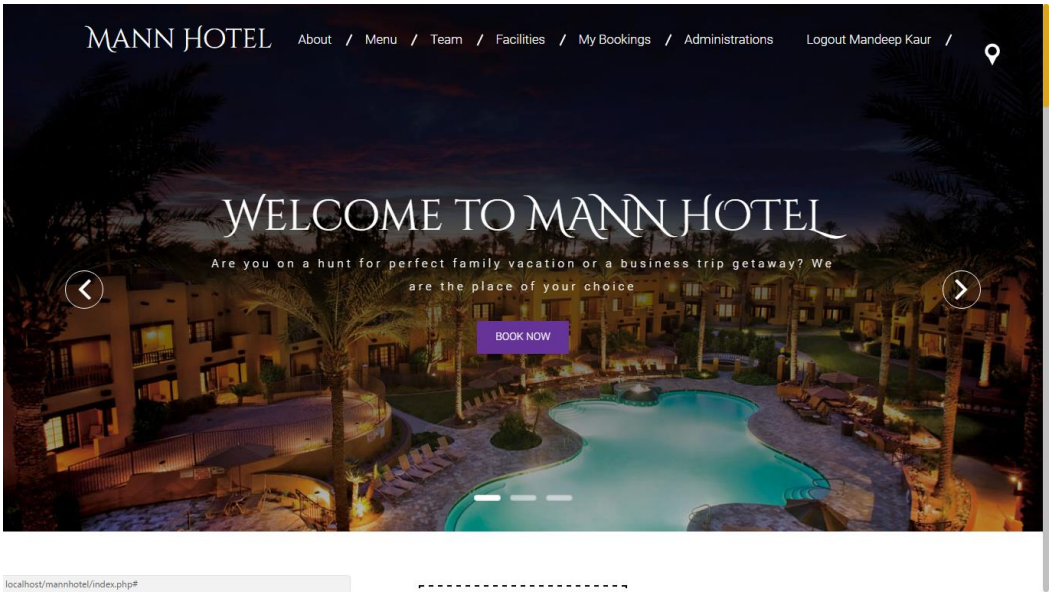


Figure 28 Homepage After Login

Login

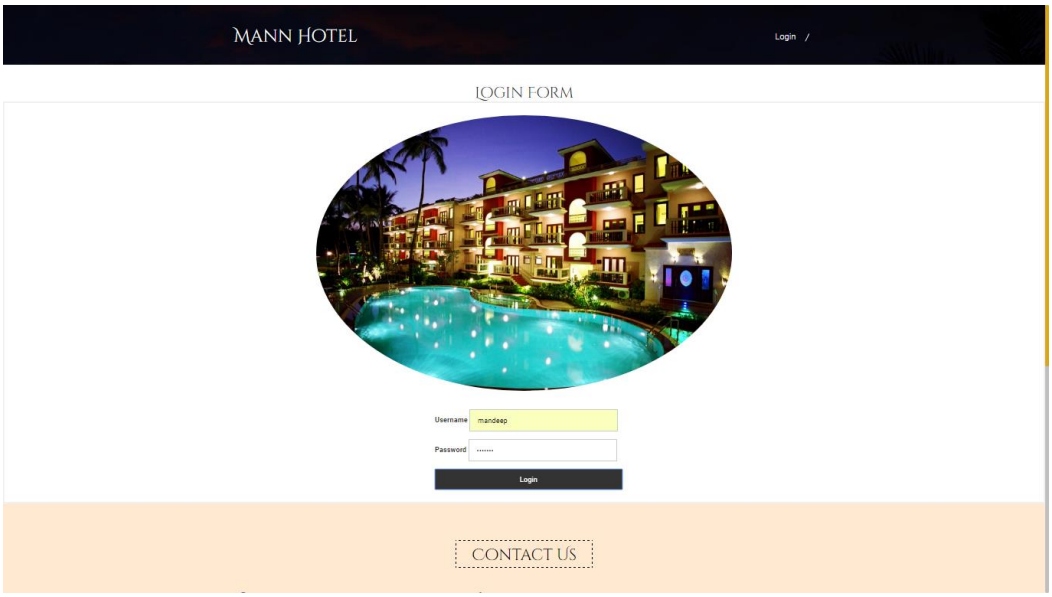


Figure 29 User Login

Bookings

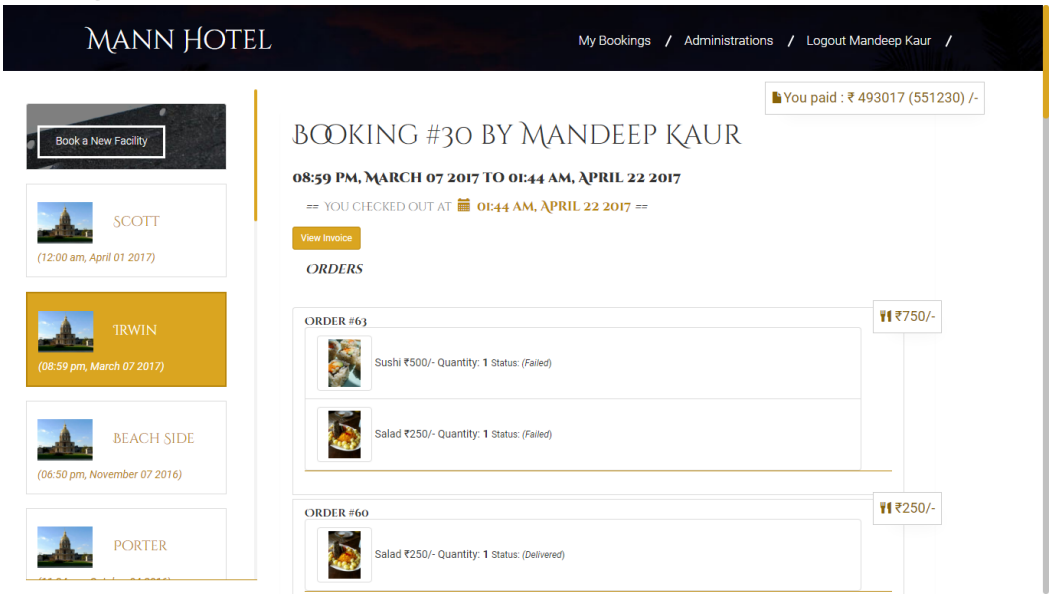


Figure 30 User Bookings

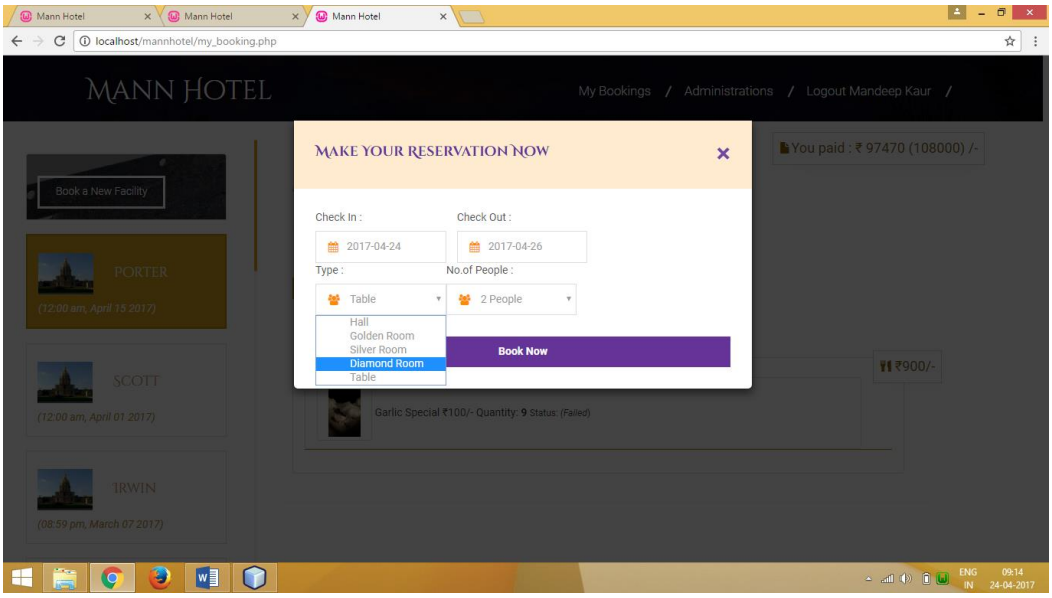


Figure 31 New Booking

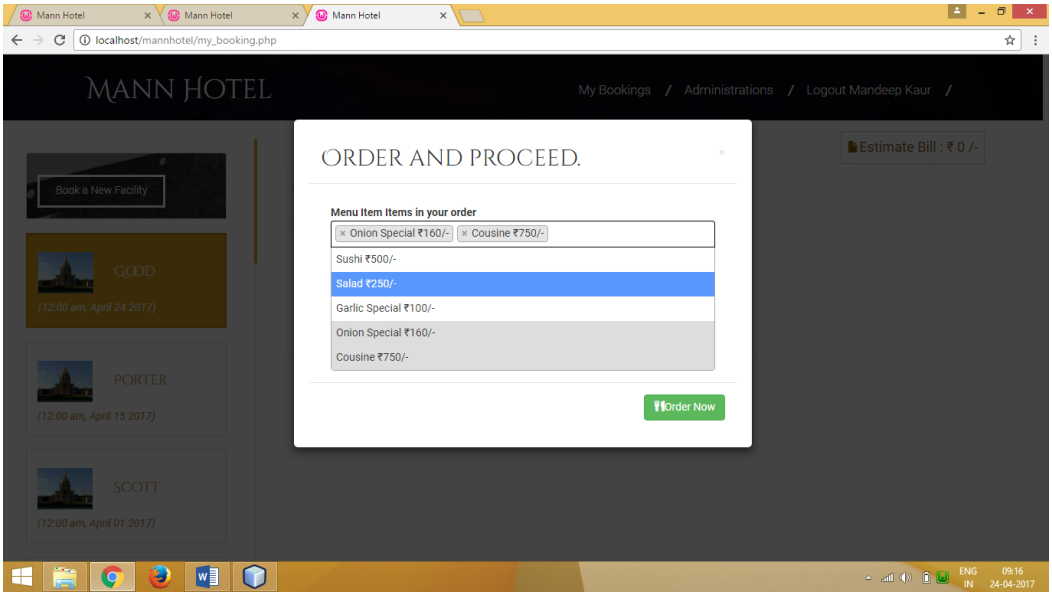


Figure 32 Booking Order

Checkout

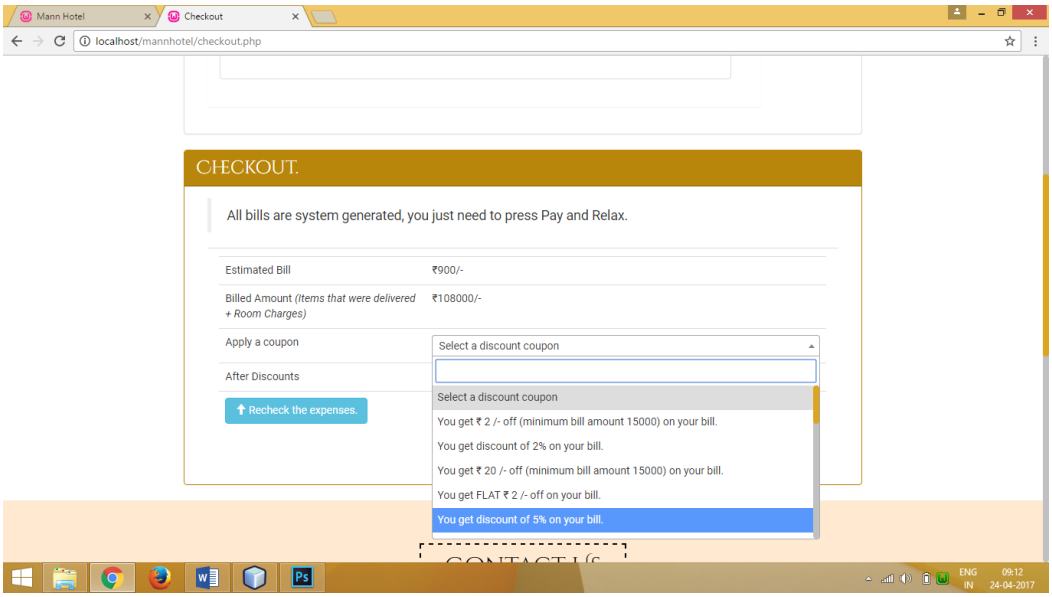


Figure 33 Checkout and Apply Coupon

Invoice Viewing and Printing

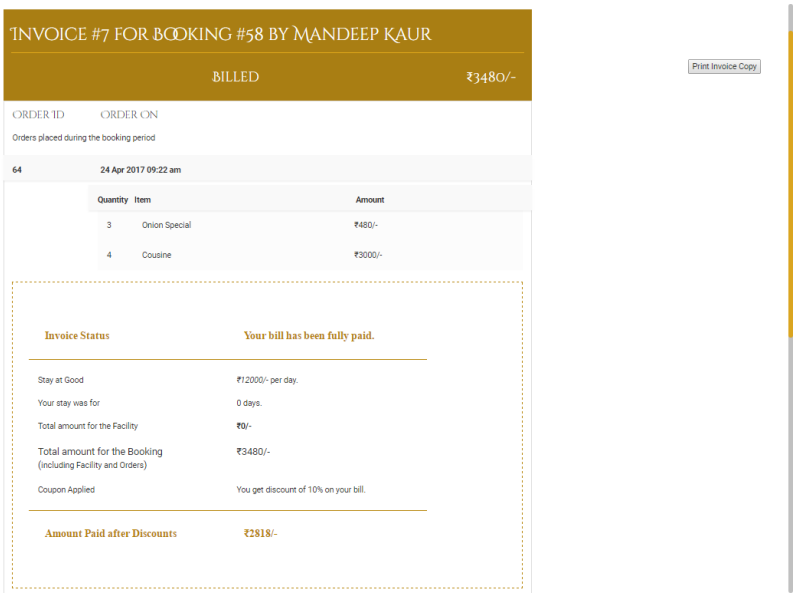


Figure 34 Invoice

Admin Side

Booking Orders View

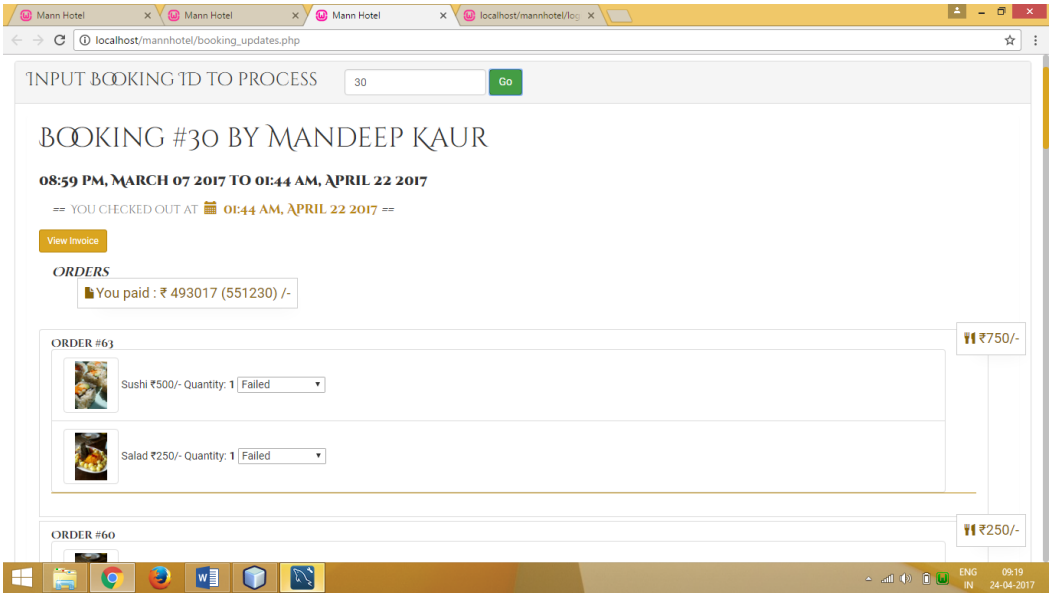


Figure 35 Booking Orders

Cost Estimation

The constructive cost model was developed by Barry W.Boehm in the late 1970s[1] and published in Boehm's 1981 book Software Engineering Economics[2] as a model for estimating effort, cost, and schedule for software projects. It drew on a study of 63 projects at TRW Aerospace where Boehm was Director of Software Research and Technology. The study examined projects ranging in size from 2,000 to 100,000 lines of code, and programming languages ranging from assembly to PL/I. These projects were based on the waterfall model of software development which was the prevalent software development process in 1981.

Basic COCOMO

Basic COCOMO compute software development effort (and cost) as a function of program size. Program size is expressed in estimated thousands of source lines of code (SLOC, KLOC).

COCOMO applies to three classes of software projects:

- **Organic projects** - "small" teams with "good" experience working with "less than rigid" requirements
- **Semi-detached projects** - "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements
- **Embedded projects** - developed within a set of "tight" constraints. It is also combination of organic and semi-detached projects.(hardware, software, operational, ...)

Mode	A	B	C	D	KLOC	Effort	Duration	Staffing
Organic	2.5	1.05	2.5	0.38	4.35	11.23	6.268	1.79

Explanation: The coefficients are set according to the project mode selected on the previous page, (as per Boehm,81). The final estimates are determined in the following manner:

effort = a*KLOCb, in person/months, with KLOC = lines of code, (in the thousands), and:

duration = c*effortd, finally:

staffing = effort/duration

Testing

Test Reports

Test Scenario	Test Description	Input	Expected Outcome	Test Result
User Login	Upon Input of Correct Username Password combination the user should be allowed to enter the system	Username Password	User Page for Logged in account	Passed
User Registeration	First time users fill out a simple form to gain access into the system	Personal Details Username Password	User Page for newly created Account	Passed
Booking Creation	Fill out Boooking details and check in	Check IN Check OUT Persons and Type of Facility	Booking Page	Passed
Order	Select Items from Menu and order them	Menu Items list	Order Placed	Passed
Checkout	Create Estimated bill for booking	Booking ID	Estimate amount of orders	Passed
Invoice	Generate invoice upon checkout	Booking Details	Task assignment to correct member and displayed in their portal	Passed
Image Upload	Upload an Image to event's gallery	Image	Image displayed in event's gallery	Passed
Image Comment	Comment on an image	Comment text	Comment shown in the comments panel of Image viewer under User's name who posted the comment	Passed
Logout	Exit the portal by ending session	Null	Secure pages require login again	Passed

System Security measures

Database / data security

Database can be secured in multiple ways and the most basic are.

- Data types for every column so wrong or error prone data never gets entered
- Using Database users
- Applying Constraints
- Use of Relation between two tables with foreign key concept rather than duplicating data to make database vulnerable towards data attacks.

It is very important for any system that the system is secure from threats like Hackers and Unauthorized edits. For that purpose as discussed earlier in this report we have provided the system with authentication levels.

User Authentication Levels

1. Guest
2. Staff
3. Admin

Guest

This user has the rights to check in, check out, place orders and print the invoices of their bookings.

Staff

Hotel staff can also perform as a guest but also they can view orders placed under any booking. Where Guests are limited to only their own.

Also they can change status of order items.

Admin

An admin has all the access of a User plus the features like deletion of rows from the database. Insertion of new venues and generating reports as well.

Reports

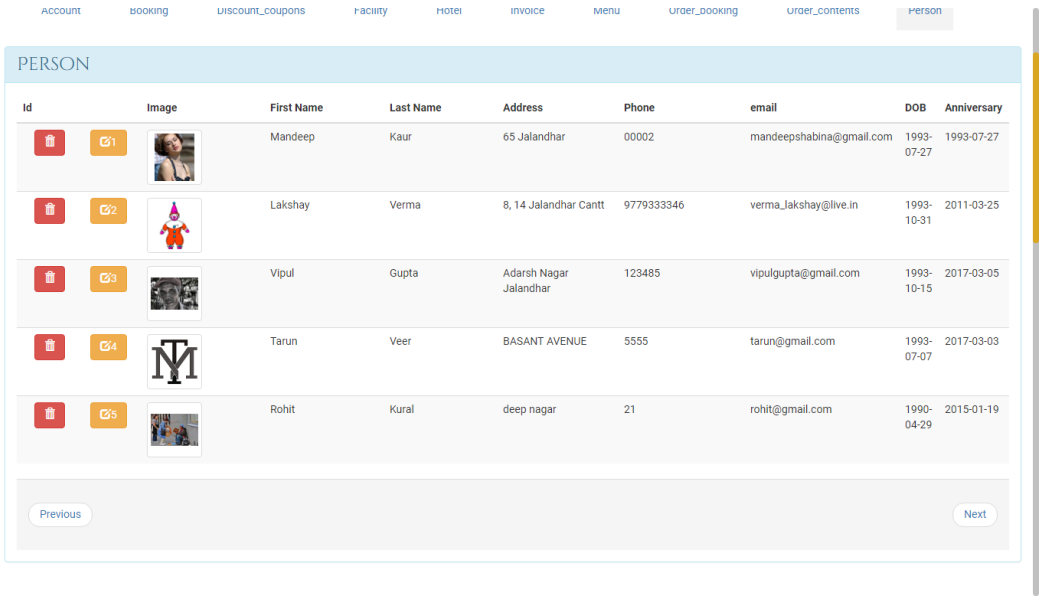


Figure 36 Person

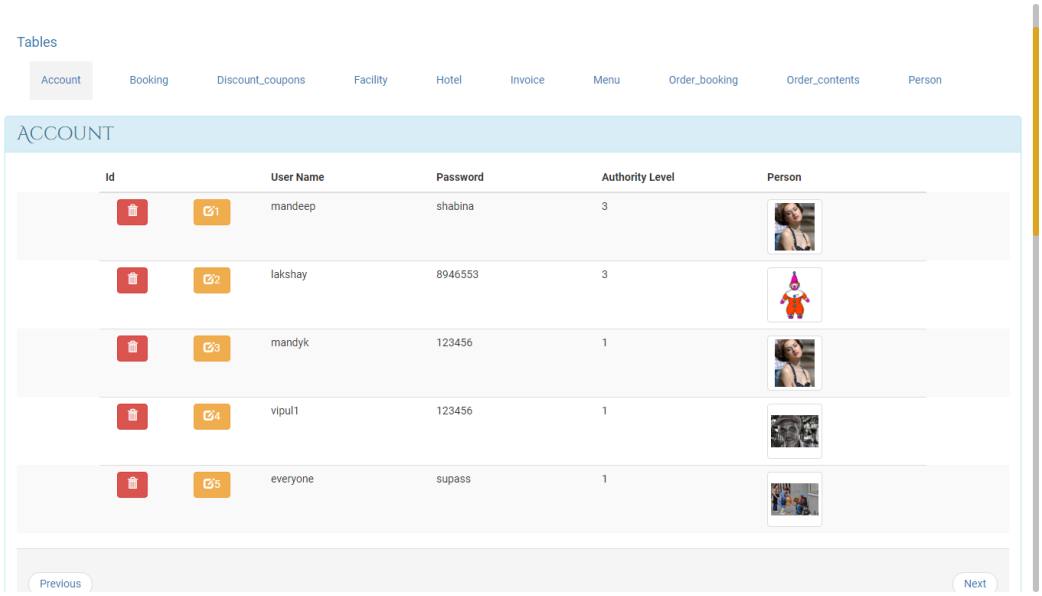


Figure 37 Account

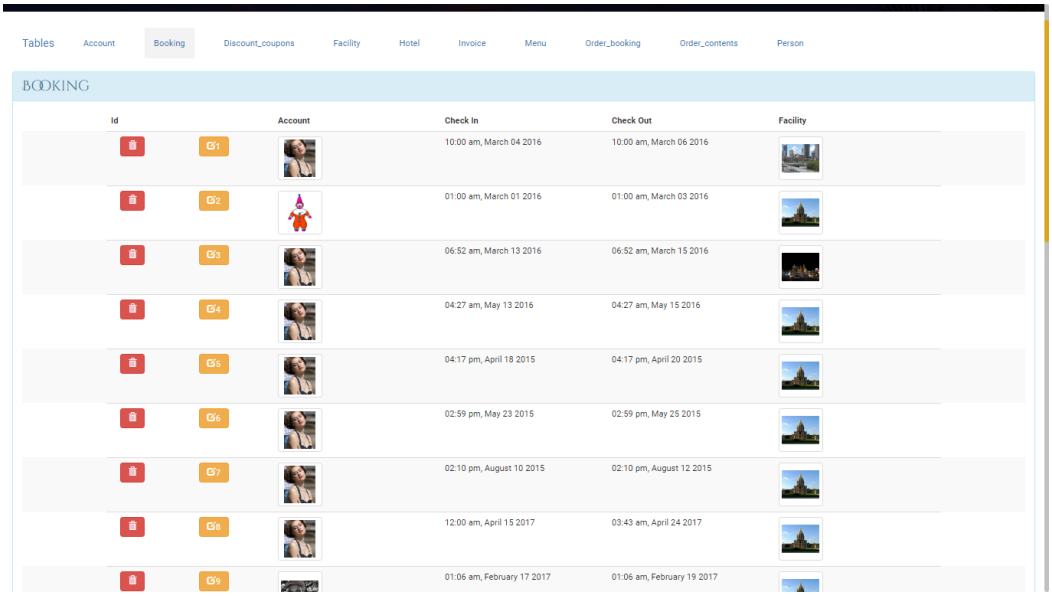


Figure 38 Booking

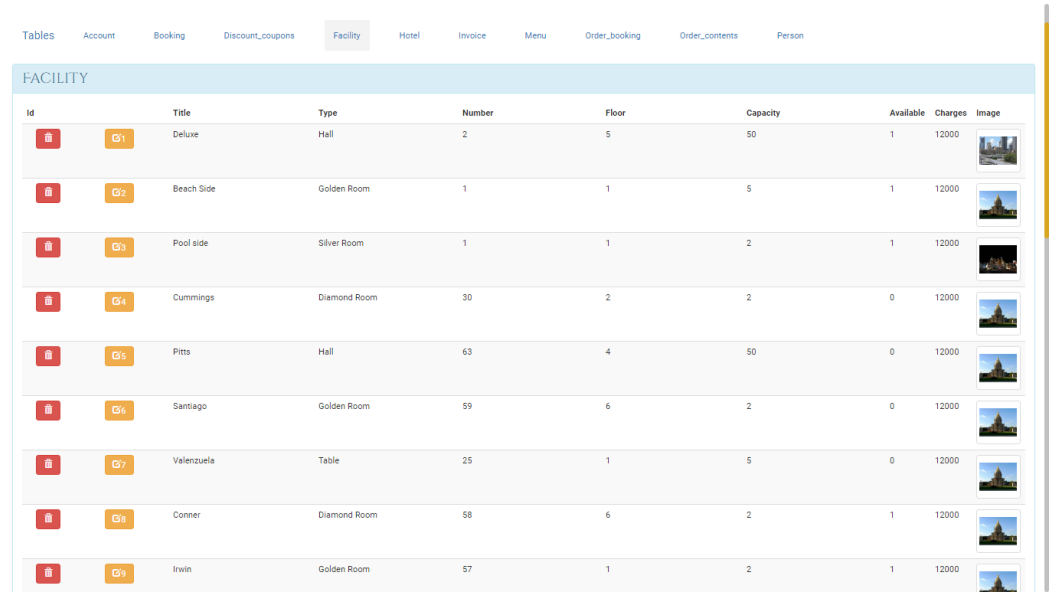


Figure 39 Facility

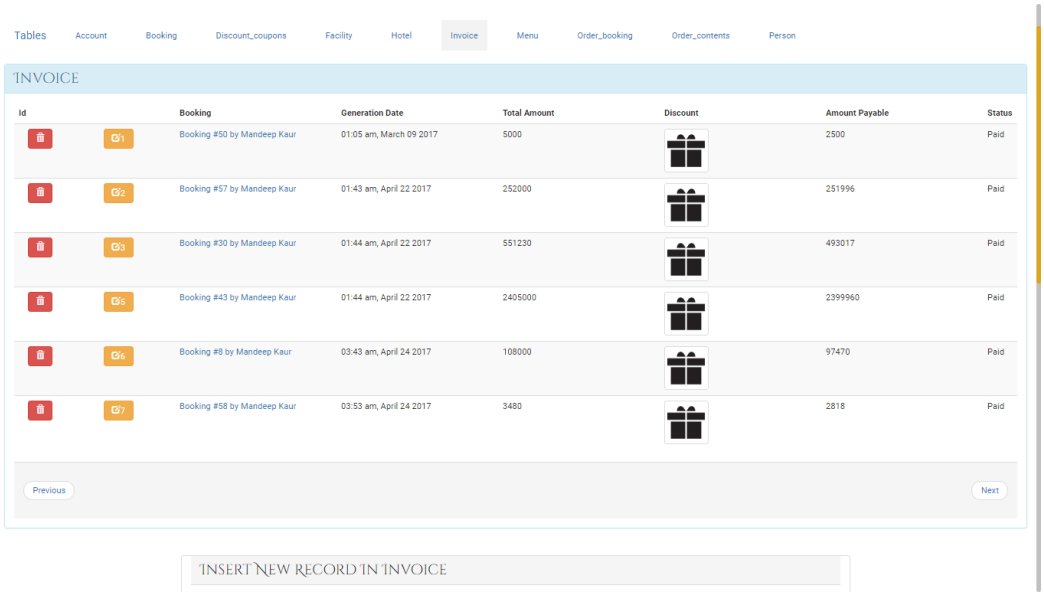


Figure 40 Invoices

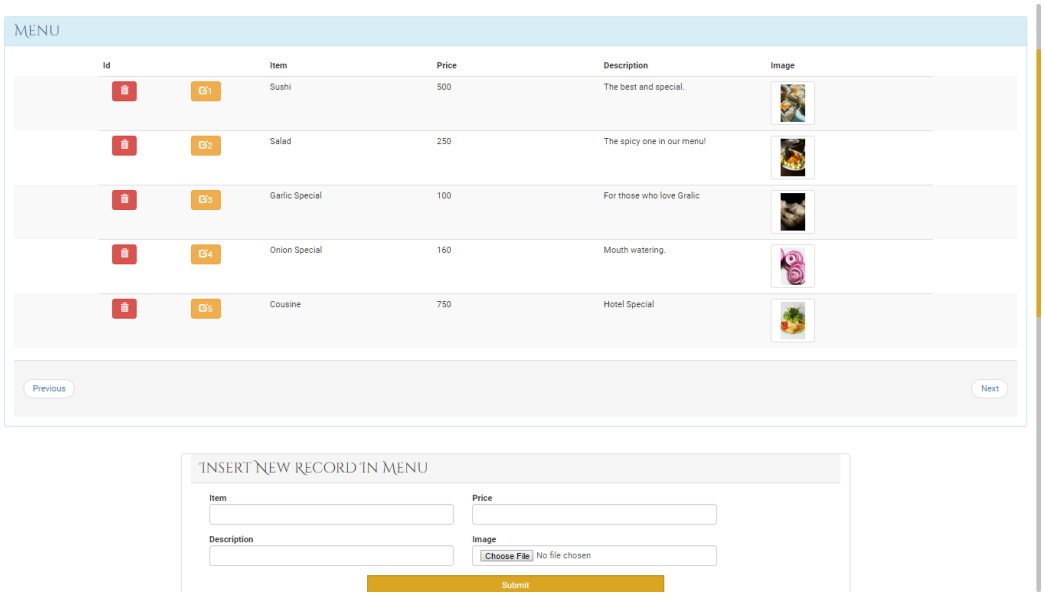


Figure 41 Hotel Menu

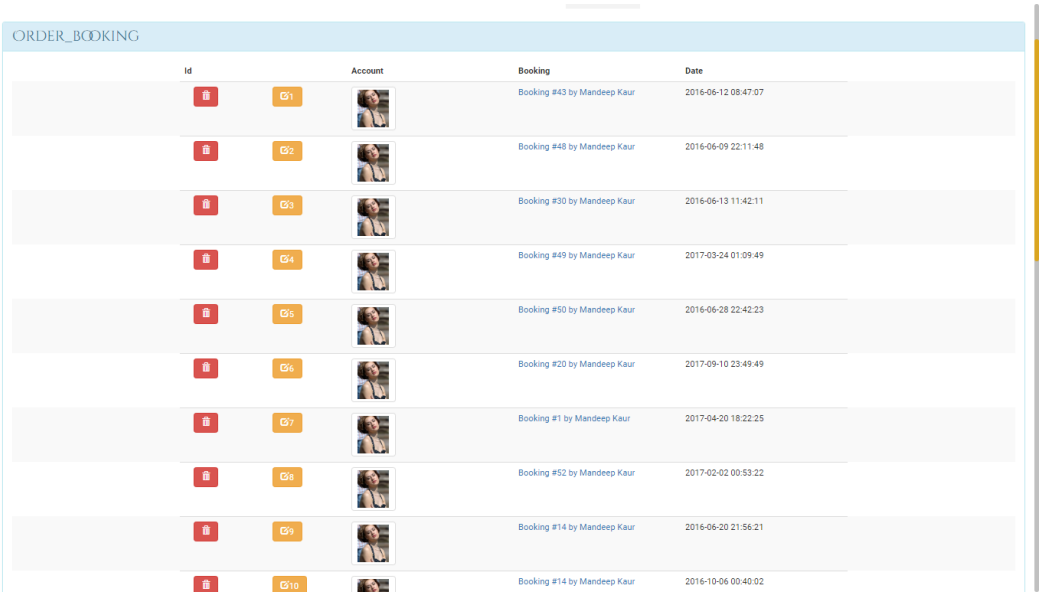


Figure 42 Order Booking

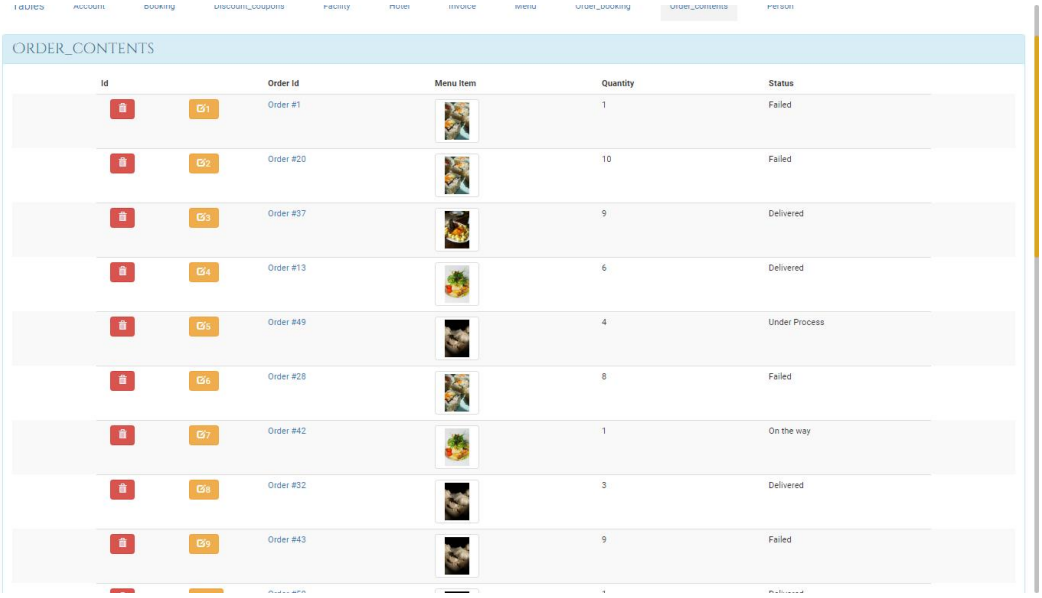


Figure 43 Order Contents

Future Scope and further enhancement of the Project

No project is 100% perfect no matter how much effort we put into making it. There always is room for improvement in a project. So is the case with this project. In this project we have made sure it has all the basic features of a Hotel Management System that could be implemented in the given time frame.

In future we can take further steps into making this system a lot more than what it currently is. Below are some of the features we can implement.

- Social media sign up
- Posting Checkin details on Facebook
- Android App for the Hotel
- Adding multiple hotels into the system to make this system into Chain of Hotels Management System
- Voice Map Navigation into the website

Bibliography

- HTML5 (Hyper Text Markup Language version 5): http://w3c.org/html.
- CSS (Cascading Style Sheet version 3): http://w3c.org/css,
- PHP: http://php.net.
- StackOverFlow (Coding Community): http://stackoverflow.com.
- The New Boston (Tutorials for Technologies Used).
- Wikipedia(The free encylopedia): http://en.wikipedia.org.
- Software Engineering a Practitioner Approach 5th Edition by Roger S. Pressman, Ph.D.
- Google (Search Engine for various Queries): http://google.co.in.
- Tool for estimating Cost : http://groups.engin.umd.umich.edu/CIS/course.des/cis525/js/f00/kutcher/kutcher.html