Project team # 6- <Online restaurant reservation system>

Team members:
Vivekananda Adepu-800967951
Disha Karunakar Shetty-800966204
Vineeth Adithya Dodla-80096657
Sanjay Tiwari-800968058

Overview:

The proposal is to create a online reservation system for a restaurant, besides providing other features like employee portal, and a job portal for the particular restaurant. The primary idea behind this project is to save data in a centralised database that can be accessed by any authorised person to get information.

Scope and objective:

This project will focus on online reservation system, employee portal and creating a job portal where individuals can look for the open job positions and drop an application for the position they are interested in. It also features a rating where customers are encouraged to post their views on the restaurant so that the restaurant can improve its functioning based on the ratings.

Project Environment:

We are developing a web based application for a restaurant. Using HTML as our primary source for developing the web page and link the data base MySQL using php interface. We have used Xampp version 5.6.14 , a free and open source cross platform web server developed by apache friends to access MySQL interface.

HIGH LEVEL REQUIREMENTS

User roles:

1. Customer: A customer is any user who can make a reservation in the restaurant for a particular day and time.
2. Employee: An employee is a person who works for the restaurant.

3. Applicant: An applicant is a user who can see the job openings available in the restaurant and apply for the same.

Initial user stories:

1. As a customer, I want to reserve a table.
2. As a customer, I want to view the menu of the restaurant so that I can see what dishes are available in the restaurant.
3. As an employee, I want to view my employee portal so that I can see my details like salary.
4. As an employee, I want to edit the menu of the restaurant so that I can add or delete dishes as per the availability.
5. As an applicant, I want to view the job openings so that I can see if there is an opening suitable for me.
6. As an applicant, I want to apply for the job openings.
7. As an employee, I want to edit the reservations so that I can make any necessary changes.
8. As a customer, I want to view the available slots.

Initial user story descriptions in order of priority:

1. As a customer I want to view available slots.
2. As a customer, I want to reserve a table.
3. As an employee(manager), I want to edit the menu of the restaurant.
4. As a customer, I want to view the menu of the restaurant.
5. As an employee(manager,parttime,fulltime), I want to view my employee portal.
6. As an employee(parttime,fulltime), I want to edit the reservations.
7. As an applicant(anyone), I want to apply for the openings.
8. As an employee, I want to edit the portal.

SPRINT 1

Story refinement, with notes:

1. As a customer I want to view available slots.

Notes:

- Rename Customer to Guest to better reflect meaning.
- Available slots refers to the availability of table in the restaurant.

Updated story:

- As a Guest, I want to view available tables in the restaurant.

2. As a customer, I want to view the menu and reserve the table.

Notes:

- Rename Customer to Guest to better reflect meaning.
- Table is reserved by providing name, contact number, number of people.

Updated story:

- As Guest, I want to view the menu and then reserve the table by providing my name, contact number, and number of people.

3. As an employee(manager), I want to edit the menu of the restaurant.

Notes:

- Employee is a general term for all employees. It may be manager, part time employee or a full time employee.
- Here edit means adding or removing dishes.
- Multiple dishes can be added or removed. So rephrase story to reflect it.
- User must logged in as Manager in order to add/remove dishes to/from the menu.
- Only the manager is given the access and right to edit the menu of the restaurant.

Updated story:

- As a Manager, I want to login to the system in order to access features specific to my role.
- As a Manager, I want to add dishes to the menu.
- As a Manager, I want to remove dishes from the menu.

Stories to be considered for sprint 1, in order of priority:

1. As a Guest, I want to view available tables in the restaurant.
2. As Guest, I want to view the menu.
3. As Guest, I want to reserve the table by providing my name, contact number, and number of people.
4. As a Manager, I want to login to the system in order to access features specific to my role.
5. As a Manager, I want to add dishes to the menu.
6. As a Manager, I want to remove dishes from the menu.

Entities and relationships:

Entities:

1. Guest
2. Employee
3. Applicant
4. Bookings
5. Menu
6. Openings

Relationships:

1. Guest reserves Restaurant_table
2. Guest views the menu.
3. Employee edits bookings.
4. Employee edits the menu
5. Applicant views the job openings
6. Applicant applies to the job openings

SPRINT1

Conceptual design

Entity: **Guest**

Attributes:

- guest_id
- Full_name
- meal_type
- phone_number
- number_of_people

Entity: **Employee**

Attributes:

- emp_id
- Password
- Name[composite]
  - first_name
  - last_name
- dob
- salary
- type
- age [derived]
- contact
- address [composite]
  - address_line_1
  - address_line_2
  - Zipcode
  - City
  - State
- date_of_joining

Entity: **Restaurant_table**

Attributes:

- table_id
- capacity
- status

Entity: **Menu**

Attributes:

dish_name
price
description

Relationship: Guest reserves Restaurant_table
Cardinality: Many to Many
Participation:

Guest has partial participation
Restaurant_table has total participation

Relationship: Guest views the menu
Cardinality: Many to One
Participation:

Guest has partial participation
Menu has total participation

Action: Manager edits the menu

Logical design

Table: **Guest**
Columns:

guest_id
Full_name
meal_type
phone_number
number_of_people

Table: **Employee**
Columns:

emp_id
Password

first_name
last_name
dob
salary
type - takes values as Manager, Full Time and Part Time
contact
address_line_1
address_line_2
Zipcode
City
State
Date_of_joining

Justification: Generalized table Employee for all kinds of employees such as Manager, Full Time and Part Time employees.

Table: **Restaurant_table**
Columns:
table_id
capacity
status

Table: **Menu**
Columns:
dish_name
price
description

SPRINT 2

Story refinement, with notes:

1. As an employee(Manager, Part Time, Full Time), I want to view my employee portal.

Notes:

- User must be logged in as an employee to access my employee portal.
- Employee portal has details about the employee name, employee id, password, date of joining, type of employee, salary, address, contact.
- Employee is categorized as Manager, Full Time, Part Time based on the type of employee.

Updated Stories:

a. As an Manager, I want to view my employee portal.
b. As a Full Time, I want to view my employee portal.
c. As a Part Time, I want to view my employee portal.

2. As an employee(Manager), I want to edit the reservations.

Notes:

- Only Manager will be able to edit the reservations of the guest.
- Rename edit to cancel.
- Cancellation of reservation is done when the guest who request for reservation did not show up after a certain time period.

Updated Stories:

a. As a Manager, I want to cancel the reservations of guests who did not show up.

3. As an employee(Manager,part time and full time), I want to edit my employee portal.

Notes:

- Rename edit to update.
- All the employees (Manager, part time and full time) can update their details viz. address , contact etc.

Updated stories:

a. As an employee(Manager, Part Time and Full Time), I want to update my details in the employee portal.

Stories in the order of priority to be considered for Sprint 2:

1. As an Manager, I want to view my employee portal.
2. As a Full Time, I want to view my employee portal.
3. As a Part Time, I want to view my employee portal.
4. As a Manager, I want to cancel the reservations of guests who did not show up.
5. As an employee(Manager, Part Time and Full Time), I want to update my details in the employee portal.

Conceptual design

Entity: **Guest**

Attributes:

- guest_id
- Full_name
- meal_type
- phone_number
- number_of_people

Entity: **Employee**

Attributes:

- emp_id
- Password
- Name[composite]
  - first_name
  - last_name
- dob
- salary
- type
- age [derived]
- contact
- address [composite]
  - address_line_1
  - address_line_2
  - Zipcode

City
State
date_of_joining

Entity: **Restaurant_table**
Attributes:
table_id
_____table_no
capacity
status

Entity: **Menu**
Attributes:
dish_name
price
description

Relationship: Guest reserves Restaurant_table
Cardinality: Many to many
Participation:
Guest has partial participation
Restaurant_table has total participation

Relationship: Guest views the menu
Cardinality: Many to One
Participation:
Guest has partial participation
Menu has total participation

Action: Manager edits the menu

Logical design :

Table: **Guest**
Columns:

guest_id
Full_name
meal_type
phone_number
Number_of_people

Highest normalization level: <4NF>

Table: **Employee**
Columns:
emp_id
Password
first_name
last_name
dob
salary
type - takes values as Manager, Full Time and Part Time
contact
address_line_1
address_line_2
Zipcode
City
State
Date_of_joining

Justification: Generalized table Employee for all kinds of employees such as Manager, Full Time and Part Time employees.

Employee Table is not in 3NF because there is transitive functional dependency between zipcode, city and state.

After Normalization:

Table: **Employee**
Columns:emp_id
Password
first_name
last_name

dob
salary
type - takes values as Manager, Full Time and Part Time
contact
address_line_1
Address_line_2
Zip_code(Foreign key references Zip_code of zipcode table)
Date_of_joining

Highest normalization level: <4NF>

Table: **city**
Columns: city_id
City_name
state_id(foreign key references state_id of state table)

Highest normalization level: <4NF>

Table: **state**
Columns: state_id
State_name

Highest normalization level: <4NF>

Table: **zipcode**
Columns: Zip_code
city_id(foreign key references city_id of city table)

Highest normalization level: <4NF>

Table: **Restaurant_table**
Columns:
table_id
table_no
Capacity

Restaurent_table is not in the 3NF normalised form.
Table_no and capacity should be treated as a separate table as there exists a transitive dependency between them.

Modified **Restaurant_table**
Columns:

Table_id
Table_no
Capacity

Highest normalization level: <4NF>

Table: table_cap
Columns:

table_id [Foreign Key, References Table_id of restaurant table]
Capacity

Highest normalization level: <4NF>

Table: table_num
Columns:
table_id [Foreign Key, References Table_id of restaurant table]
Table_num

Highest normalization level: <4NF>

Table: **Menu**
Columns:

<u>dish_name</u>
price
Description

Highest normalization level: <4NF>

Table: **Reservation**
Columns:

<u>Table_id</u>
<u>date_booked</u>
guest_id [foreign key; references guest_id of Guest table]

Justification: guest_id of Reservation table references guest_id of Guest table.

Highest normalization level: <4NF>

SPRINT 3

Story refinement, with notes:

1. As a User, I want to view the job openings.

Notes:
- Rename job openings to vacancies available at the restaurant.
- Vacancies can be multiple for different positions like Manager, Part Time and Full Time.

Updated Story:
a. As a User, I want to view the vacancies available.

2. As a User, I want to apply for the vacancies available.

Notes:
- A User must be logged in to apply for job openings. Add user story for sign-up, login, logout.
- Once the User has sign-up he/she becomes an applicant.
- An applicant should submit his/her details while applying for the job.
- An applicant can check the status of the application.

Updated user stories:
a. As a user I want to sign-up so that I can access features specific to my role.
b. As a user I want to login so that I can apply for vacancies by providing my details.
c. As an applicant I want to check the status of my application.
d. As an applicant, I want to logout.

Stories in the order of priority to be considered for Sprint 3:

1. As a user I want to view the vacancies available.

2. As a user I want to sign-up to create my applicant account.
3. As an applicant I want to login so that I can apply for vacancies by providing my details.
4. As an applicant I want to login to check the status of my application.
5. As an applicant, I want to logout.

Conceptual design

Entity: **Guest**
Attributes:

- <u>guest_id</u>
- Full_name
- phone_number
- number_of_people

Entity: **Employee**
Attributes:

- <u>emp_id</u>
- Password
- Name[composite]
    - first_name
    - last_name
- ssn
- dob
- salary
- type
- age [derived]
- contact
- address [composite]
    - address_line_1
    - address_line_2
    - Zipcode
- date_of_joining

Entity: **Restaurant_table**
Attributes:

table_no
capacity

Entity: **Menu**
Attributes:
dish_name
price
description

Entity: **Applicant**
Attributes:
Applicant_id
Name[composite]
first_name
last_name
date_of_birth
contact
address [composite]
address_line_1
address_line_2
zip_code

Date_of_application
Qualification

Entity: **Applicant_Signup**
Attributes:
user_name
_____password

Entity: **Vacancy**
Attributes:
id
_____title
num_of_vac

Relationship: Guest reserves Restaurant_table
Cardinality: Many to Many
Participation:
  Guest has partial participation
  Restaurant_table has total participation

Relationship: Guest views the menu
Cardinality: Many to One
Participation:
  Guest has partial participation
  Menu has total participation

Relationship: applicant applies for the vacancy
Cardinality: Many to Many
Participation:
Applicant has partial participation
Vacancy have total participation

Action: Manager edits the menu
Cardinality: Many to one
Participation:
Manager has a partial participation
Menu has total participation

Logical design:

Table: **Guest**
Columns:
  <u>guest_id</u>
  Full_name
  phone_number
  Number_of_people

Highest normalization level: <4NF>

Indexes:

Index #: <clustered>

Columns: <guest_id>

Justification: Guest_id uniquely identifies each row in the table and is used the most to query this table. Example if we want to see the reservation of a particular guest we can use the guest_id.

Table: **Employee**

Columns: emp_id

Password

first_name

last_name

dob

salary

type [foreign key, references type of employee_type table]

contact

address_line_1

Address_line_2

Zip_code(Foreign key references Zip_code of zipcode table)

Date_of_joining

Justification: Generalized table Employee for all kinds of employees such as Manager, Full Time and Part Time employees.

Indexes:

Index #: <clustered>

Columns: emp_id

Justification: emp_id is a unique attribute that determines a particular employee. The index emp_id on the table employee will be a clustered index as it is an index on the primary key. There is no need to specify it is unique since it is a primary key.

Indexes:

Index #: <non-clustered>

Columns: type

Justification: type is also used to query this table often. like to find out total number of people in a particular role.

Highest normalization level: <4NF>
Table: **Employee_ssn**
Columns:
emp_id [foreign key references emp_id of employee table]
ssn

Justification: Generalized table Employee_ssn which is used to identify each employee.

Highest normalization level: <4NF>

Indexes:
Index #: <clustered>
Columns: emp_id ,ssn
Justification: emp_id, ssn is the primary key which uniquely identifies each row and hence is the clustered index here.

Table: **city**
Columns: city_id
City_name
state_id(foreign key references state_id of state table)

Highest normalization level: <4NF>
Indexes:
Index #: <clustered>
Columns: <city_id>
Justification: city_id is used most frequently to query this table and is also its primary key and hence forms a clustered index here.

Table: **state**
Columns: state_id
State_name
Indexes:
Index #: <clustered>
Columns: <state_id>
Justification: Justification: state_id is used most frequently to query this table and is also its primary key and hence forms a clustered index here.

Highest normalization level: <4NF>

Table: **zipcode**
Columns: Zip_code
city_id (foreign key references city_id of city table)
Indexes:
Index #: <clustered>
Columns: <zipcode`>
Justification: Justification: zipcode is used most frequently to query this table and is also its primary key and hence forms a clustered index here.

Highest normalization level: <4NF>

Table: **Restaurant_table**
Columns:
table_no
Capacity
Indexes:
Index #: <clustered>
Columns: <table_no>
Justification: Justification: table_no is used most frequently to query this table and is also its primary key and hence forms a clustered index here.

Highest normalization level: <4NF>

Table: **Menu**
Columns:
dish_name
price
Description

Highest normalization level: <4NF>

Indexes:

Index #: <clustered>

Columns: dish_name

Justification: dish_name is the most frequently used attribute to query this table. Dish_name here is a primary key and thus the index dish_name is a clustered index.

Table: **Reservation**

Columns:

reservation_id

table_number

date_booked

guest_id [foreign key; references guest_id of Guest table]

time

Justification: guest_id of Reservation table references guest_id of Guest table.

Indexes:

Index #: <clustered>

Columns: reservation_id

Justification: Reservation_id is used frequently to query the table reservation. Reservation_id is already unique since it forms primary key of the reservation table. Hence it forms a clustered index.

Indexes:

Index #: <non-clustered>

Columns: date_booked

Justification: This table can be queried using the date_booked column frequently to get the number of reservations on a particular date. Hence it forms a good non clustered index.

Highest normalization level: <4NF>

Table: applicant_signup

Columns:

user name

Password

Indexes:

Index #: <clustered>

Columns: user_name

Justification: This table is mostly queried using user_name during login process.

Highest normalization level: <4NF>

Table: **Applicant**

Columns:

<u>applicant_id</u>

first_name
last_name
dob
salary
openings - takes values as Manager, Full Time and Part Time
contact
address_line_1
address_line_2
Zip_code (Foreign key references Zip_code of zipcode table)
Date_of_application
position [foreign key references type of employee_type]
qualification
email_id
user_name [foreign key references user_name of applicant_signup table]

Indexes:

Index #: <clustered>
Columns: applicant_id

Justification: applicant_id is used frequently to query the table to get information about a particular applicant.

Indexes:

Index #: <non-clustered>
Columns: user_name

Justification: user_name is used to query the table to get information about the applicant and to join the table with other tables. As they don't affect the physical order of the table they can be classified as non clustered index.

Highest normalization level: <4NF>

Table: **Applicant_ssn**
Columns:

applicant_id [foreign key references applicant_id of applicant table]
ssn

Justification: Generalized table Applicant_ssn which is used to identify each applicant
Highest normalization level: <4NF>

Indexes:

Index #: <clustered>
Columns: applicant_id ,ssn

Justification: applicant_id, ssn is the primary key which uniquely identifies each row and hence is the clustered index here.

Table: **employee_type**
Columns: type

Highest normalization level: <4NF>

Indexes:

Index #: <clustered>
Columns: type

Justification: type is used frequently to query this table and is also its primary key and hence forms the clustered index here.

Table: **vacancy**
Columns: id
title
num-of_vac

Highest normalization level: <4NF>

Indexes:

Index #: <clustered>
Columns: id

Justification: id is used frequently to query this table and is also its primary key and hence forms the clustered index here.

Indexes:

Index #: <non-clustered>
Columns: title
Justification: title can be used frequently to query this table to determine vacancies for a particular role and hence forms the non-clustered index here.

Stored programs:

Stored function:<rest_reservation>
Parameters:<capacity, date of booking, time>
Goal: The purpose of the function rest_reservation is to make a reservation for the guest according to the information given by the guest such as number of people,date and time of the booking and return the table_id reserved for that particular guest.

Stored procedure:<add_applicant>
Parameters:<appl_id>
Goal: This procedure is used to add details of an applicant to the employee table after he has been appointed for a particular position by the Manager.

Stored procedure:<view_status>
Parameters:<appl_id,status>
Goal: The view_status procedure is used to display the status of an applicant by passing his applicant id, showing if he has been accepted for the role he applied for or not.

Trigger:<update> on <insert>

Goal: The sole purpose of this trigger is to update the number of vacancies remaining for a particular position after a applicant is accepted for a particular role.

Event:<delete reservation>

Goal: The purpose of creating this event is to delete a reservation where the booking date has elapsed [limit is 1 day]. Delete reservation event will remove the records in the reservation table where the booking date is less than the current date.