

**CMPE 138 / 180-38 : Database System 1**

**Wedding Planner Application System**

**Submitted to:**

**Dr. Kong Li**

**Date of Submission:**

**11/30/2016**

**Submitted by: Team #13**

**Anushree Sinha [010444031]**

**Polar Halim [008646534]**

**Ritu Singh [010828714]**

**Vimmi Swami [010811840]**

**Fall 2016**

TABLE OF CONTENTS

TABLE OF FIGURES

1. Project Background

For the term project regarding this class, we have decided to create an application that use relational management to perform its application. The project information are listed below:

* Project Title: Wedding Planner Application System
* Database Engine: MySQL Version 5.7.15
* Framework: Bootstrap, HTML5, CSS
* Languages: Java, JSP

1. Overview

The primary focus of this project will be to apply the knowledge of relational database management. In this project, the function of create, update, delete, count, etc will be implemented. On top of the database management, bootstrap is also used to create a more user friendly website. With the project completion progress spanning for around two months, most of the planned goal has been achieved with several adjustment made. Also, further improvement is always possible to improve the performance and functionality of the application.

1. Database Design

In order to design a good application, a solid architecture should be design first. For this project, we decided to map every single components and the relation by drawing it in the Entity Relationship Diagram. Below are the components along with its relationship.

* 1. ER Diagram
     1. Proposed ER Diagram



* + 1. Updated ER Diagram

Untitled Diagram.png

* + 1. ER Diagram Changes

From the two ER diagrams above, it can be seen that there are several changes made after the proposed ER diagram. Below are the list of changes made:

**Minor Changes**

Minor changes includes renaming of attributes in most of the entities so that it is more relatable between the database tables and ER Diagram.

**Major Changes**

* Customer Entity:
  + Email\_id is changed to be unique to prevent customers in creating a duplicate account or double account.
  + Wedding date is included in customer instead of wedding entity, as it is more relevent to keep the customer details intact in customer itself and be used as foreign key in wedding.
* Wedding Entity

The cardinalities associated with wedding\_component is changed. Reason being, the changes in wedding component entity and accordingly the relation between them.

* Guest Entity:

-guest entity is changed from strong entity to weak entity with g\_id as partial key because a guest depends totally on a wedding. It can’t exist without a wedding alone.

* Email is made as a unique key to allow same email to repeat for a different wedding and disallow same email for same wedding and hence accordingly cardinality is changed with wedding entity.
* Attendance attribute is removed as it will require every guest to be enrolled in the application leading to unnecessary management of a data which needs periodic check for its validity in system to retain it.
* Wedding component entity:
* Wedding component is made as a strong entity instead of weak entity with wc\_id as key, due to many reasons. Major influence was in normalizing the table in database which resulted in many joins and unnecessary tables. As major and critical part of application depends upon querying this table, simplicity and performance was highly required. Making it strong entity, resolved the issues with lesser number of rows in the table (and for a wedding) and lesser number of joining with other tables.
* Since a single row of wedding\_component is designed to have all the other components selected by user for a single wedding, the cardinality is changed accordingly with wedding entity.
* In decorator, venue, performer, photographer, and caterer entities:
  + A key attribute, web\_address, is added to these entities to allow user to view the options of components and then make a selection with based on preference.
  + The application is made with the constraint of choosing at least one component, the venue entity is made as a requirement so that it will allow the application to create an invitation to the guests.
    1. Reasons Of Updation
* Having wedding component as weak entity was creating multiple entries for a single wedding, in other words, multiple entries for a single wedding id; and that would have been a bad design notion in terms of storage and normalization. Hence, it was made a strong entity to have a single entry with all the chosen component for each wedding.
* The attributes for each of the tables related to wedding component such as photographer, performer etc. were further extended to include web\_address column to aid customers in selection.
* Guest has been made a weak entity to ensure that same guests can come in more than one wedding.
  1. Table Design
     1. Functional Dependencies

In customer table, c\_id uniquely identifies the rest of the attributes. As multivalued attributes are not allowed for normalizations, contact\_no attributes are made as a separate table. Below is the FDs diagram

Please note, the member table in an application point of view table for storing login details and attribute password is to store pwd details for Event organizer

As you can see in below diagram, we do not have any transient FDs, all tables, have all attributes being uniquely identified by the primary keys.

fd1

fd2

* + 1. Normalization

The breakdown can be seen from the points below:

1NF:

Since we have multivalued attribute in Customer table-contact\_no, we bring the table to 1NF form by making separate table for problem attribute.Thus we have CONTACT\_NUMBER a separate table with c\_id and c\_no as composite primary keys. With above, all the tables in database have a primary key,no multivalued or composite attributes and have all non key members depending on respected Primary keys.

* All the non-key attributes of all the tables present in this wedding planner database are dependent on primary key, hence all are in 1 NF.

2NF:

With above schema in 1 NF, as we can see from FD diagram above,

all non-prime attributes depends on all attributes of a composite primary key (no partial dependency) and are fully dependent on primary key (no partial dependency).

The only case where we have composite primary keys are Checklist and Guest. Even in those, composite primary keys together, uniquely identifies rest of the table and not individually.

3NF:

With above with explanation in 2NF form, and based on our FD diagram, we can see we do not have any transient FDs and any partial dependencies in any table. Hence, the database is already in 3NF form.

refrence

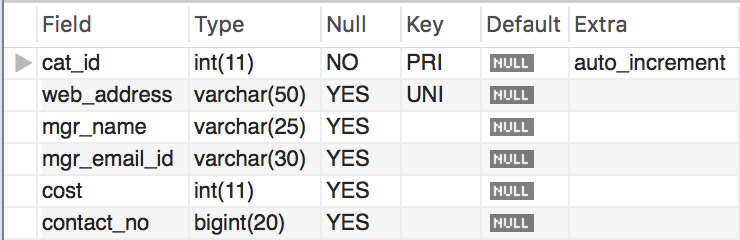
* + 1. The Normal Form

At the end of the normalization process, all of the tables in the database is brought to 3NF form.

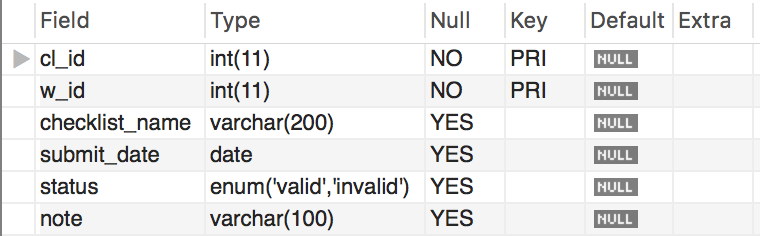
* + 1. Specification of each DB

The only database used is MySQL. The DB Specification are shown below:

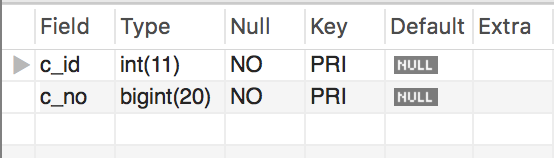
Caterer



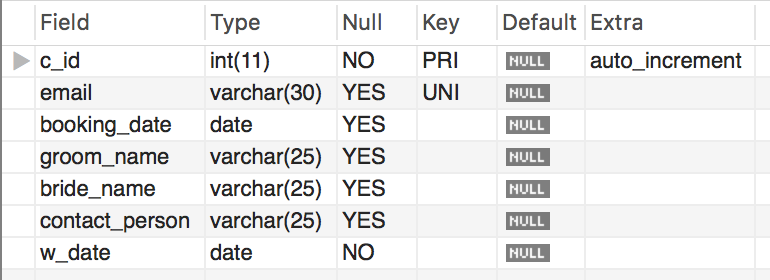
checklist



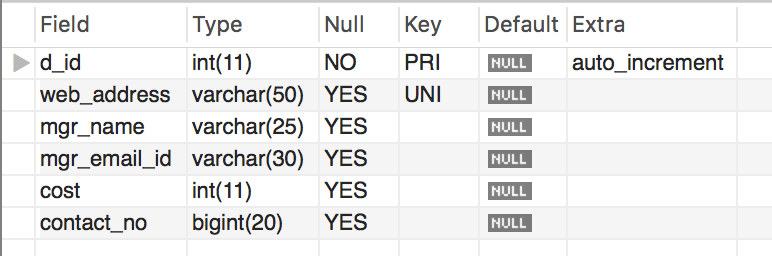
contact\_no



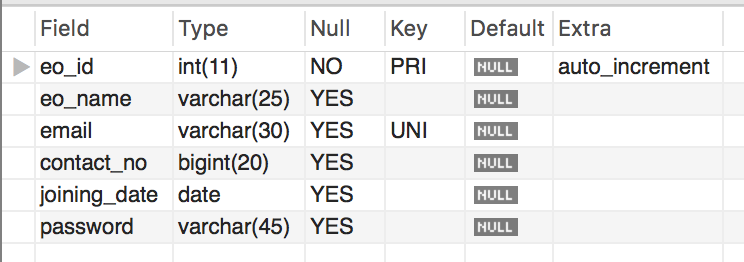
customer



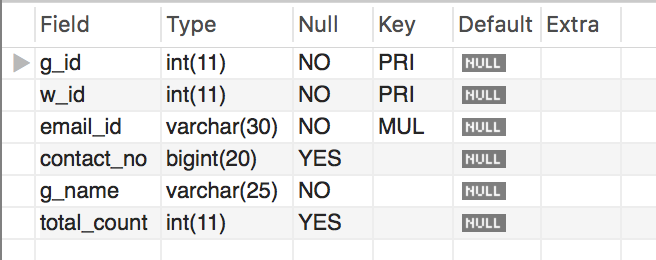
decorator



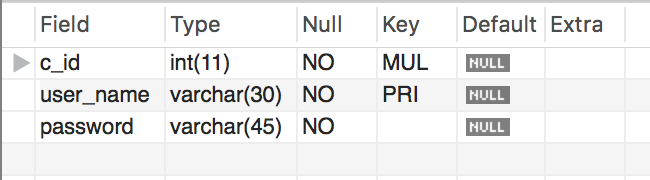
event\_organizer



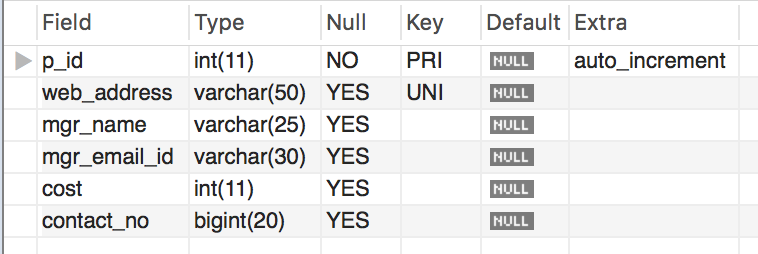
guest



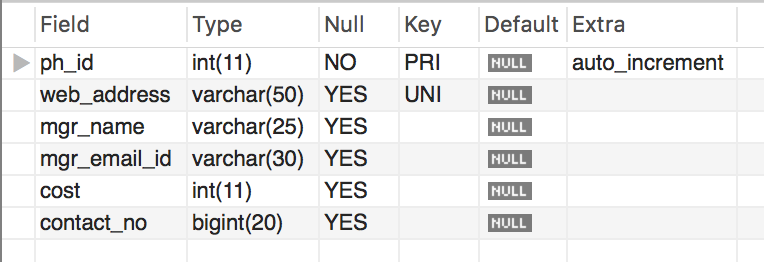
member



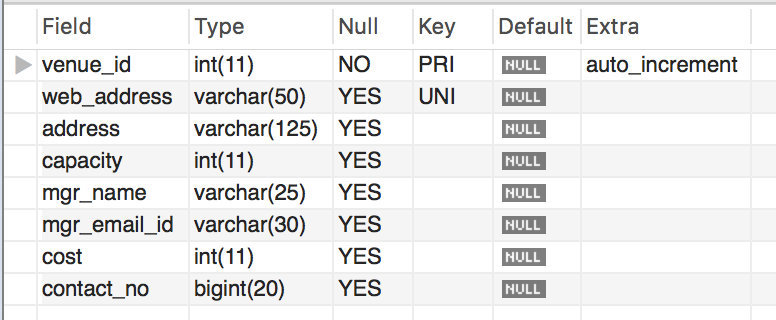
performer



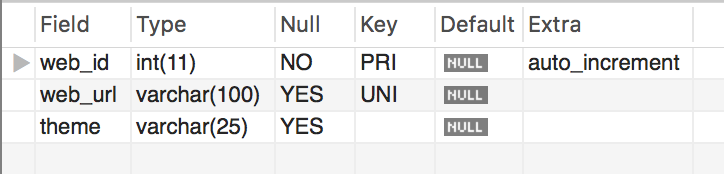
photographer



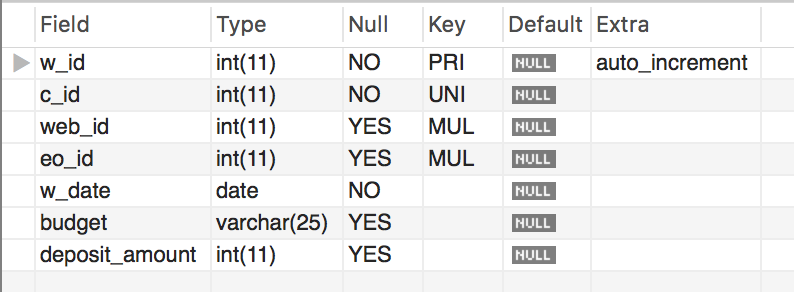
venue



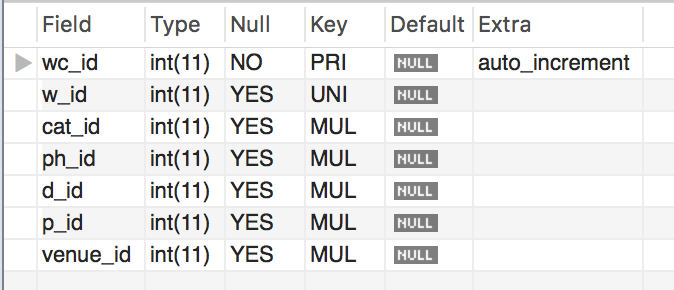
website



wedding



wedding\_component

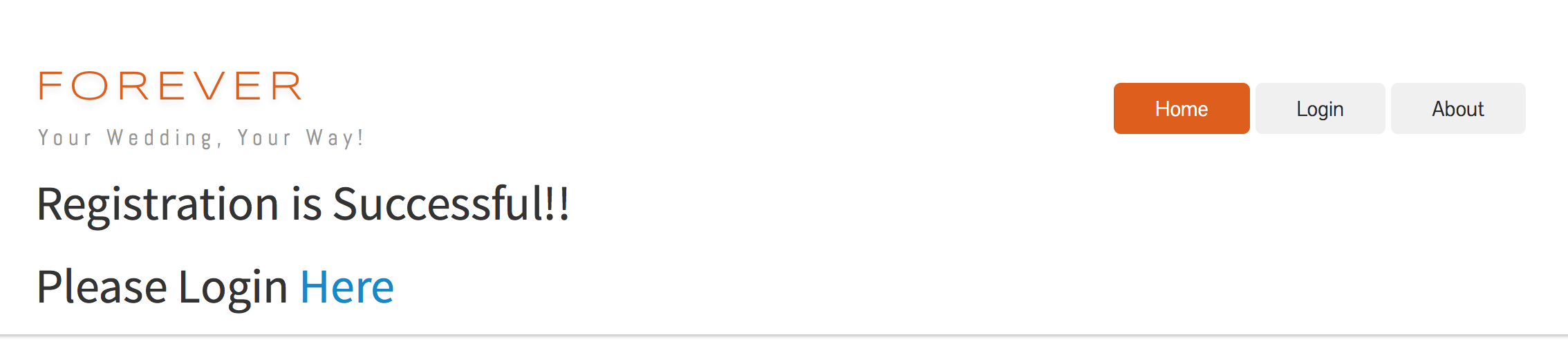


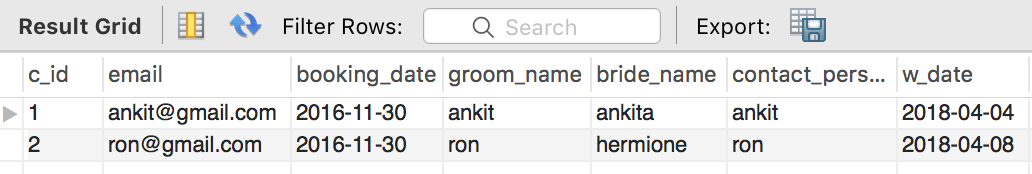
* + 1. Transaction

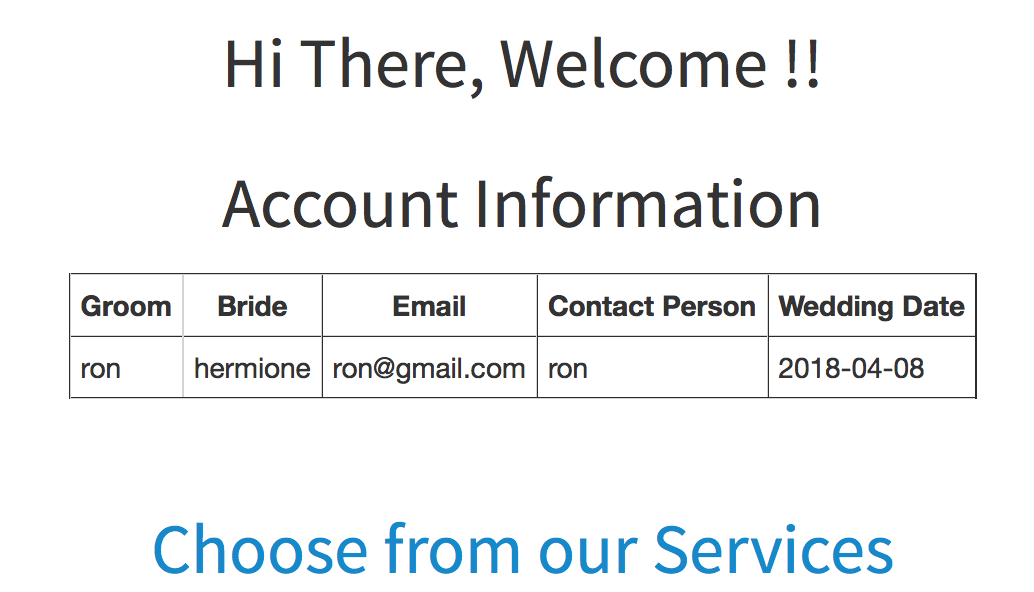
No example as no transaction used explicitly.

* + 1. **Sample Execution for Database and Web Application**

1. Registration

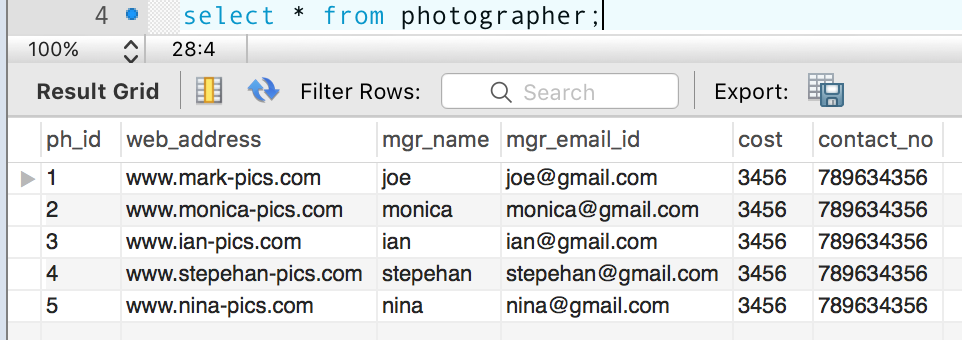




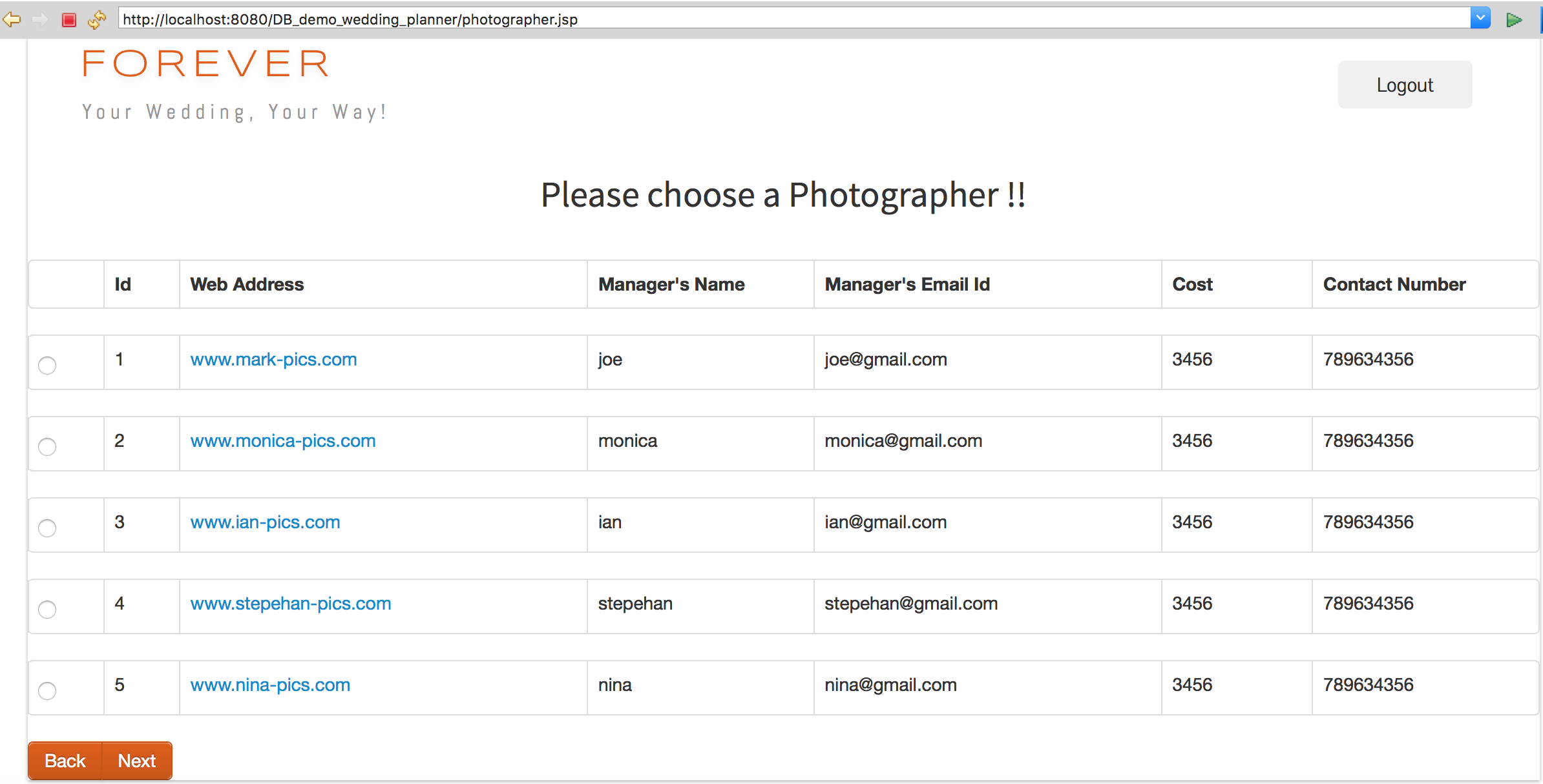


b. Selection

Static Reference tables to choose wedding components:

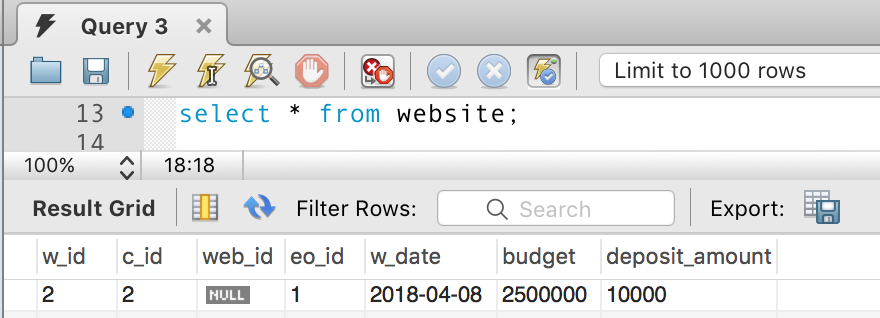


Options comes on Web UI for customer selection:

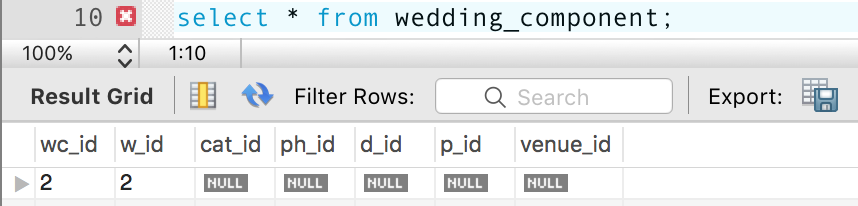


Wedding Component table keeps track of the selection made by the customer.

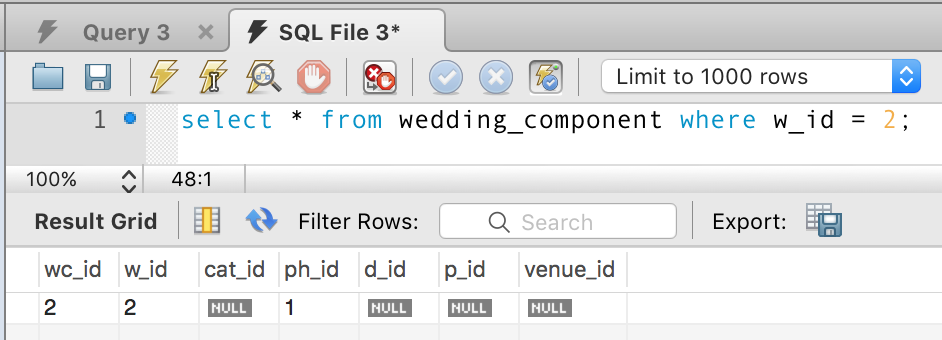
In this case, for customer name = “Ron”, customer ID is 2 and wedding ID is 2



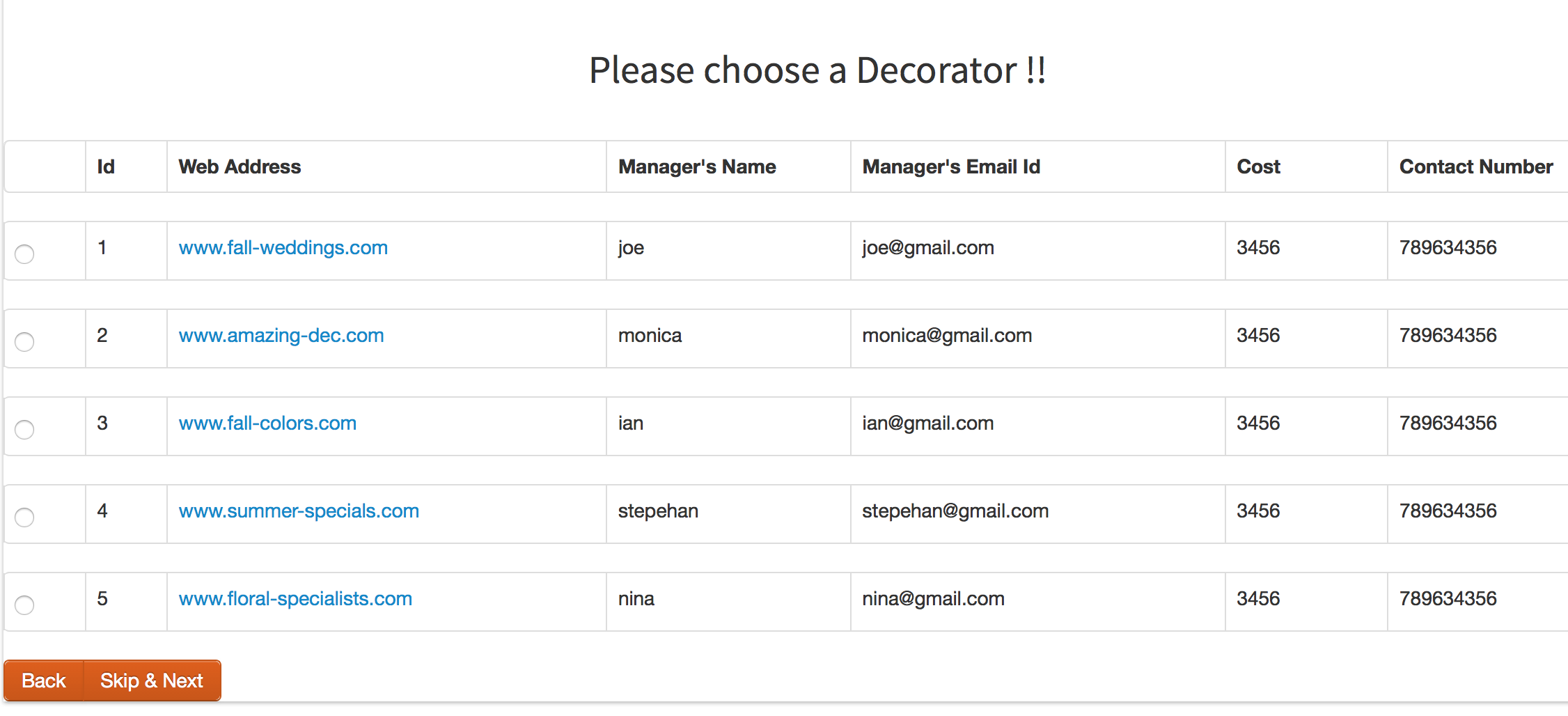
Initially there wouldn’t be any selection for this customer c\_id=2 in the wedding\_component table :

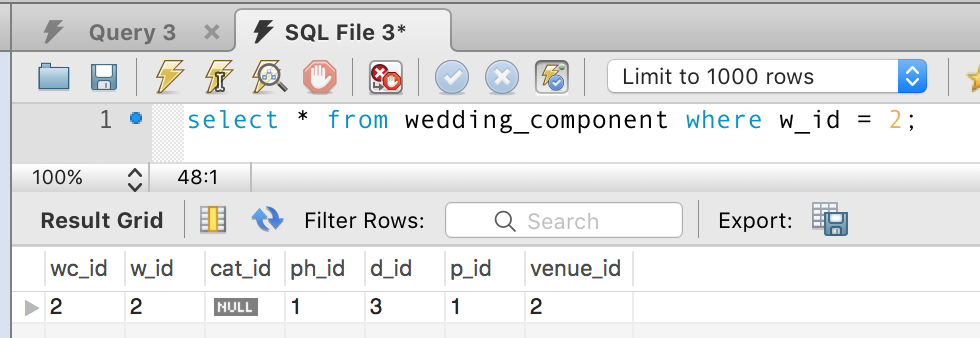


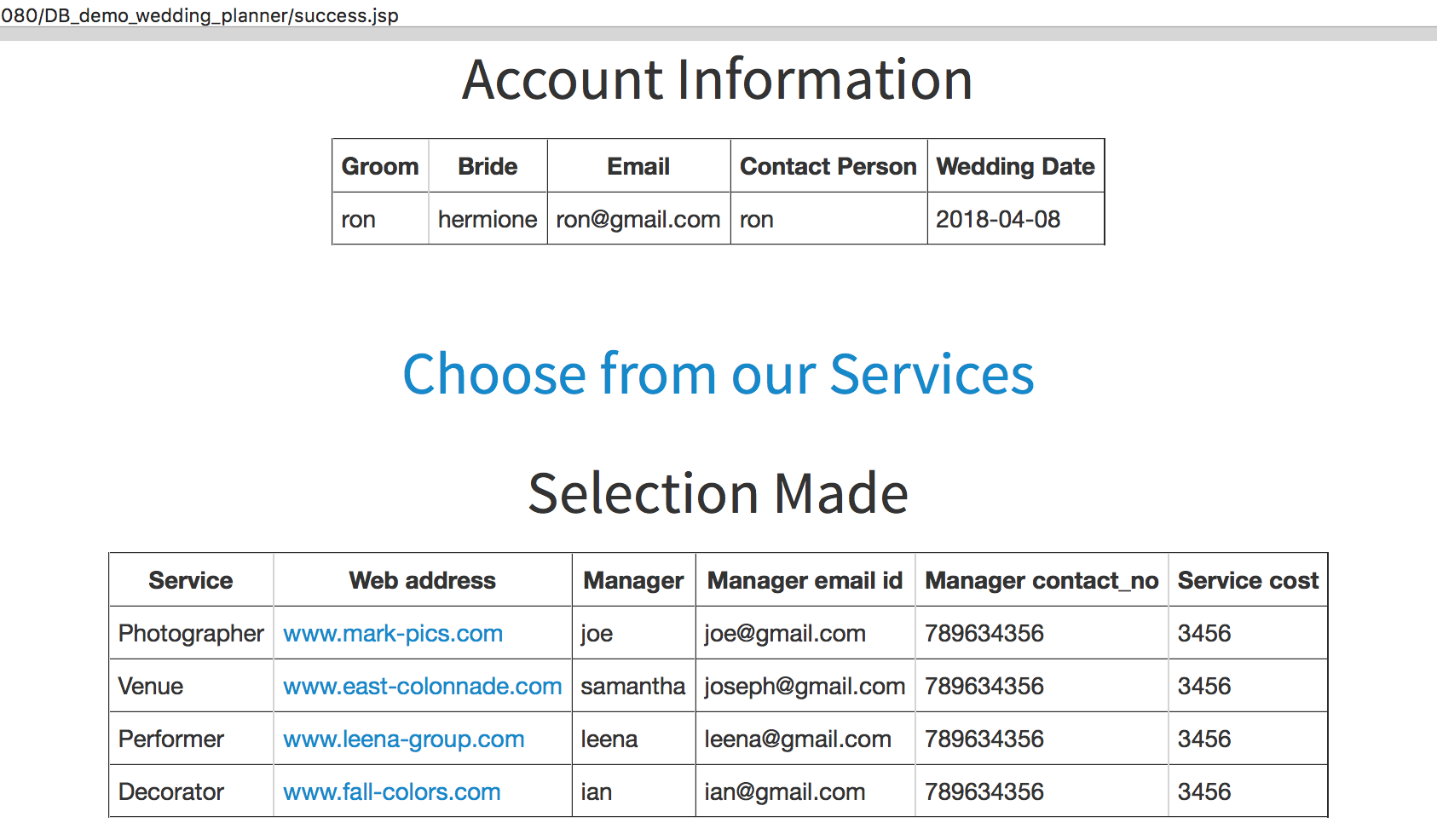
Once the customer select a photographer, the photographer id - ph\_id gets updated in the wedding component table and this photographer ID becomes unavailable for the customer choosing photographer for the same wedding date.



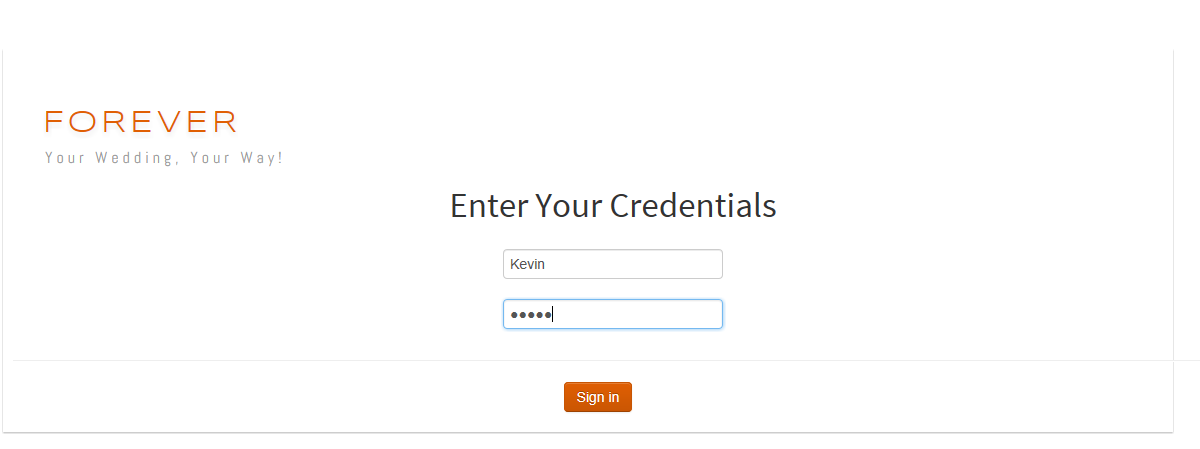
Similarly, the customer chooses or skips the selection of decorator, caterer and performer. Venue selection has been made mandatory to put the address on the invitation website.



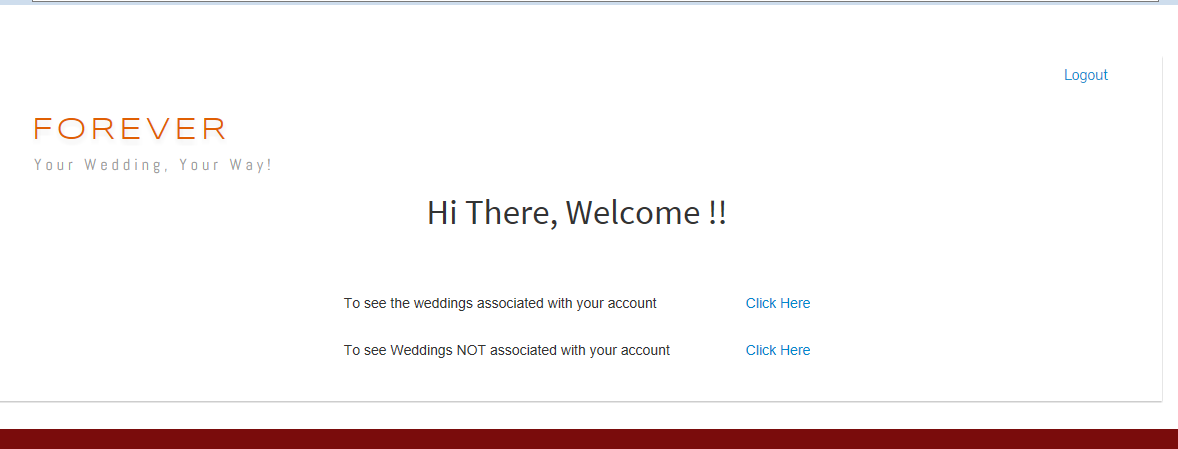




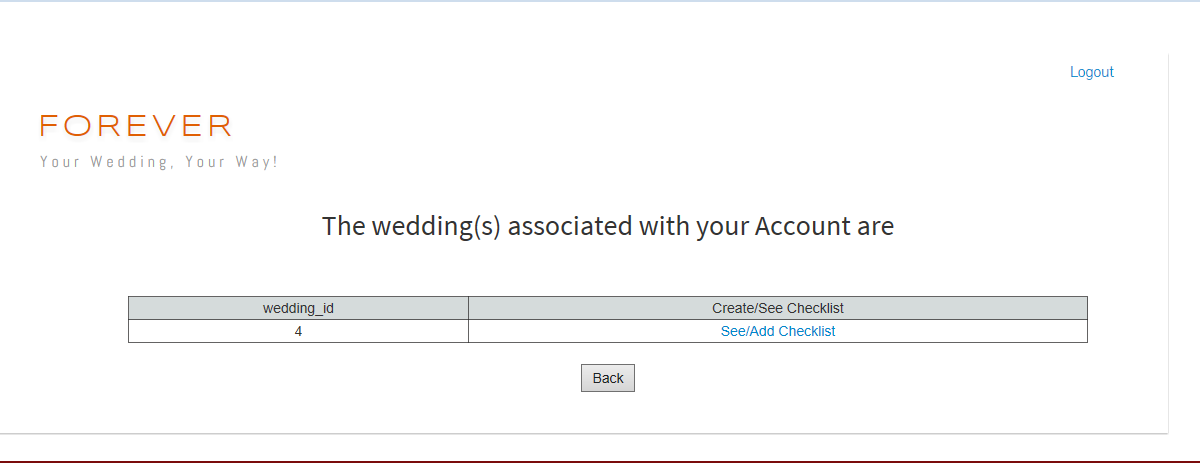
Event Organizer Login Page



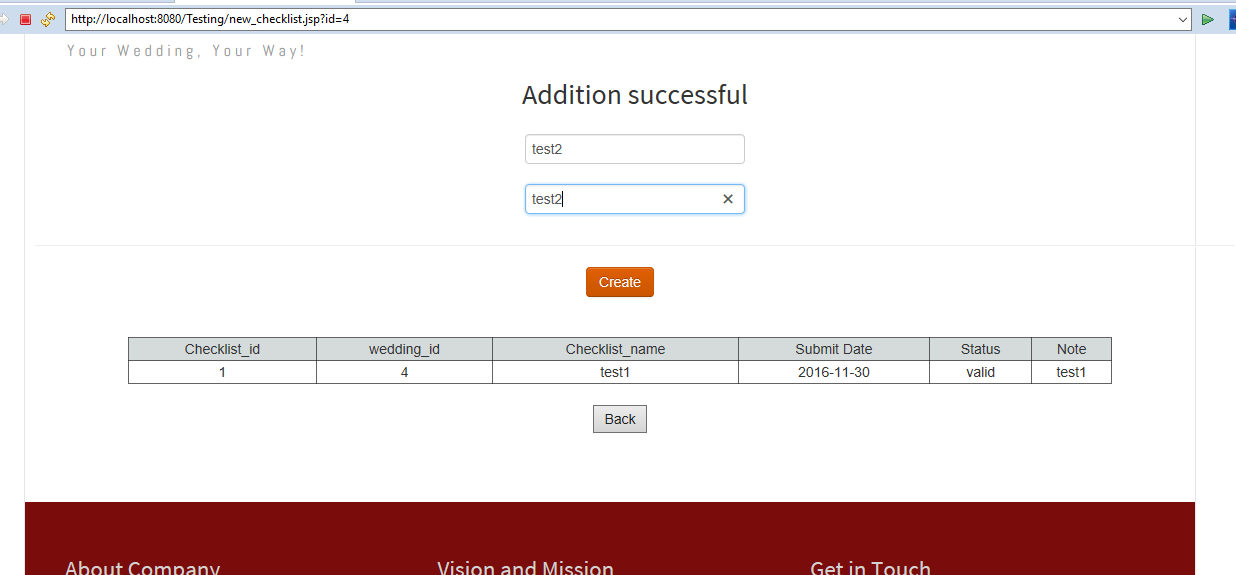
Event Organizer home page after successful login



On clicking on the link saying wedding associated with your account, the following page appears -



On clicking See/Add Checklist -



The checklists already associated with the wedding appears and also we get an option to create new.

* 1. Database Application Design
     1. Implementation

-User registers himself with desired details.

-In backend entries are made for that client in wedding, wedding\_component, members and customer tables.

-on successful registration, user is redirected to login page.

-on successful log in, user is redirected to success page/home page

-Home page shows customer details, service chosen(if any), link to add services and if any service chosen, then link to add guests.

-customer add/skips services such as photographers, decorators, performers, caterers and venue one by one based on their availability on his wedding date.

In the end, customer is redirected to home page

-if atleast one service chosen, link to add guests is activated.

Customer clicks on the link and add/deletes guests.

-On top, User is also given option to view his customized Invite.

Event organizer- An event organizer is assigned to a wedding by the administrator. He monitors the status of the wedding and creates checklists for the weddings assigned to him.

* + 1. Transaction

All the database connections are closed in the source code after the queries are executed.

1. Key Functionality

* Allowing user to create to choose the wedding components and other options to chose from based on the wedding date.
* User can also create a list of guests and manage their invitations and count in one page.
* Available options are based on the availability of components based the existing bookings and user’s wedding date.
* Event Organizers are assigned to take care of all the groundwork.

1. Flow Of Data:

* Registration:

1. Customer registers himself by filling data:

Customer table: Insertion of new row for the user data,c\_id created automatically.

Member table: filled with user name, password and c\_id.

Wedding table: a row is created with c\_id and wedding date(w\_date), w\_id created automatically.

Wedding\_Component table: create a row with w\_id from above, wc\_id is created automatically.

Have w\_id, c\_id, wc\_id, w\_date are added to the session.

Customer is not allowed to enter same email id , and most of the entries are mandate.

2. On successful registration, user is pointed to LogIn.

3. On successful LogIn, user is redirected to homepage

User is shows his own details.

Given link to select services/components one by one on click of radio button or customer can skip a service.(Venue is selected as default if skipped).The selection ids are updated in wedding\_component table for his w\_id.

Services are shown based on their availability based on his wedding date.

User is redirected to homepage on completion of service selection and shown his service selections.

User can re choose services from previous link and will be updated as per then available choices based on his wedding date.

Once a single service is chosen and updated in wedding\_component table, link for adding guests.

On click of the link, user is redirected to another page for adding guests.

Any addition of guests is shown below in the page the table guest is populated with details for the w\_id.

g\_id is generated by selecting max(g\_id)+1 for the w\_id in guest table if any entry already exist. Else g\_id is made 1 for the w\_id.

User can now delete the added guests.

User is also shown a link to his personalized invitation.

On click of this link, user can view his invitation with chosen venue.

Use can go back or simply click on page’s “Forever” text to go back home, or logout from homepage.

4. Event Organizer:

Event organizers are registered from backend as its company related procedure.

Each EO log in from Log in page through separate link given for EOs.

On successful log in, EO can view weddings assigned to him by administrators.

For each wedding assigned, EO can view the existing added checklists and can add new checklists through a link.Checklist table is populated with data for a given w\_id. Cl\_id creation is based on same algorithm as g\_id.

Deletion of checklist is by administrator by validating the completion of the checklist.

EO can logout using logout button.

1. Major Design Decision and Trade-offs

We have made a single table wedding component as strong entity. Each wedding id will have single wedding component id wc\_id and single row in wedding\_component id will have component selection details. As except for venue, other components are optional for user, high chances are that few may be NULL. Ideally, this can be a bad design based on the company performance., ie, if the company doesn’t have customers choosing their offered components choices, they will not be chosen and left NULL.

If we normalize it , we’ll have many tables made( one to restrict id to a specific component, eg id 1 = performer, 2= caterer, etc, other table for each wedding component for each wedding to be mapped to these ids selected, eg, wc\_id 1 will have id 1, 4, wc\_id 2 may have id 2,3 etc, other table to have wedding and wedding component related. Hence minimum three tables. As we can understand, we have a dependent entity involved and many tables interrelated. As we have most important section of choosing these components based on the wedding date and availability of the components, most of the queries will be running on these tables join. Considering such complexities of queries from database point of view and performance requirement from business point of view, both are poor. Hence we decided to keep the single table as strong entities with possible null values to ease the search processing with performances with only disadvantage of memory wastage.which can be dealt ig the business is good and less nulls are inserted.

1. Minor Modification
2. Major Modification
3. Unique Design

The whole project is unique design from db schema level to application’s registration to component selection to invitations and guests.

The uniqueness lies in no availability of sources for reference and hence the whole project is completely based on the team hard work, creativity and database concepts learnt in class. Several highlights are:

* Multi-user assistance
* Various E-R concepts included - multi-valued attributes, weak entities
* User friendly & **aesthetic** web UI

1. Task Breakdown

|  |  |  |
| --- | --- | --- |
| **Team Member: Anushree Sinha** | | |
| **Project Pages** | **Major Tasks** | **Completion Date** |
| log.jsp | * Validates the username and password entered by the user | Oct 21,2016 |
| reg.jsp | -Takes care of the registration part. Records all the data entered by the user | Oct 28,2016 |
| eo\_login.jsp | -Validates the username and password entered by the event organizer | Oct 31,2016 |
| eo\_home.jsp | -Home page of the event organizer. Helps him to redirect to pages where he can see the wedding ids attached to his account. | Nov 2,2016 |
| add\_checklist.jsp | * Here the event organizer gets an option to add a checklist for the wedding ids attached to his account. A checklist shows the progress of the wedding preparations. | Nov 5, 2016 |
| eo\_new\_wedding.jsp | * This page shows the Event organizer the wedding ids which are not associated to his account. | Nov 10,2016 |
| new\_checklist.jsp | * Here the event organizer can see the checklists already attached to the wedding id and he can also fill in a form on this page to create a new checklist. | Nov 17,2016 |
| Login.java | * This file validates the username and password entered by the user by in log.jsp by querying the table - “member” | Oct 21,2016 |
| Register.java | -The details entered by the user on reg.jsp are stored in the customer table by using this file. | Oct 28,2016 |
| EO\_login.java | * The username and password entered by the event organizer are validated by querying the “event\_organizer” table. | Oct 31, 2016 |
| EO\_wedding.java | * This file retrieves the wedding ids associated with the event organizer by querying the table - “wedding”. | Nov 5,2016 |
| EO\_new\_wedding.java | * This file retrieves the wedding components of the wedding ids not attached to the event organizer by querying tables - “wedding” and “wedding\_component” | Nov 10,2016 |
| Checklist.java | -This file helps in displaying the existing checklist components and inserting new checklists for the wedding ids by querying table - “checklist”. | Nov 17,2016 |

|  |  |  |
| --- | --- | --- |
| **Team Member: Polar Halim** | | |
| Project Pages | Major Tasks | Completion Date |
| WEB UI | * Create a responsive webpage * Bootstrap + HTML5 + CSS * Enhance the look and feel of the webpage | Oct 21, 2016  Beautify again by modifying template:  Nov 7, 2016 |
| about.jsp | * Page to display the major contribution | Nov 28, 2016 |
| gallery.jsp | * Display a sample product of our application | Nov 15, 2016 |
| guest.jsp | * Create a unique web url for each invitation by retrieving data from the table | Nov 26, 2016 |
| index.jsp | * Main page of the website * Redirection and flow of the webpage | Oct 30, 2016 |
| invitation.jsp | * Retrieve the name of the bride and the groom to be displayed * Retrieve the venue from the db table | Nov 20, 2016 |
| logout.jsp | * Clear all the session hold | Nov 3, 2016 |

|  |  |  |
| --- | --- | --- |
| **Team Member: Ritu Singh** | | |
| Project Pages | Major Tasks | Completion Date |
| photographer.jsp | -Provides list of available photographers for the specific wedding date to customer to choose from.  -This selection has been kept optional. | Oct 20, 2016 |
| ph\_update.jsp | Updates the wedding component table with desired selection of photographer. | Oct 23, 2016 |
| decorator.jsp | Provides list of available decorators for the specific wedding date to customer to choose from.  -This selection has been kept optional. | Oct 25, 2016 |
| de\_update.jsp | Updates the wedding component table with desired selection of decorator. | Oct 28, 2016 |
| caterer.jsp | Provides list of available caterers for the specific wedding date to customer to choose from.  -This selection has been kept optional. | Nov 5, 2016 |
| ca\_update.jsp | Updates the wedding component table with desired selection of caterer. | Nov 10, 2016 |
| performer.jsp | Provides list of available performers for the specific wedding date to customer to choose from.  -This selection has been kept optional. | Nov 12, 2016 |
| pf\_update.jsp | Updates the wedding component table with desired selection of performer | Nov 16, 2016 |
| venue.jsp | Provides list of available venues for the specific wedding date to customer to choose from.  -If a customer skips this selection, a default venue is assigned to him to aid in publishing wedding invitation website. | Nov 22, 2016 |
| ve\_update.jsp | Updates the wedding component with desired selection of venue. | Nov 24, 2016 |
| ve\_update\_default.jsp | Updates the wedding component with default venue if the customer skips the selection. | Nov 28, 2016 |

|  |  |  |
| --- | --- | --- |
| **Team Member: Vimmi Swami** | | |
| Project Pages | Major Tasks | Completion Date |
| guest.jsp | -allow addition of guests for a wedding id  -disallow duplicate email ids(guests)  -show existing guests(if any)  - allow deletion of guests(if any) | Nov 26, 2016 |
| guestud.jsp | -implement guest deletion (a row of guest shown through guest.jsp) | Nov 26, 2016 |
| guest.java | -implement addition of guests on click on ADD | Nov 26, 2016 |
| success,jsp | -welcome customer  -show customer details.  -provide link to add components/services  -show components/services chosen(if any)  -show link to add guests only if atleast one service is selected, hidden otherwise.  -provide link to add guests if above fulfilled. | Nov 26, 2016 |

1. Test Plan Execution

* Register user with valid details.
* Register users with already used email id
* Register user with a date previous to current date
* Once registered:
* Add components with valid choices
* Re choose components and check if they are updated with recent choices.
* Skip venue selection.
* Add another user with same wedding date and check if already taken components are not shown in options.
* Login as Event Organizer and create checklists for the wedding ids attached to the account.

1. Further Improvements

In every project, there is always room for improvement. Just like all other projects, improvement for our project are:

* Auto invitation sending to guest once the guests are finalized
* Disallow any changes after a certain date
* Gift registry
* Payment Processing
* Feedback Column
* Rating Feature

1. Project Postmortem

This project revolves around the entities involved in wedding planning; their interactions, dependencies and relationships in real world. Active actors involved in this are customers, event organizers and passive ones are photographers, decorators, performers and caterers. Active actors either register themselves with the wedding planning service (customers) or are already a part of system offering service to customers (event organizer).

Static entries for the wedding components like photographers, decorators, performers, venue and caterers are put into reference tables to be available for selection by customers.

At backend database level, the

1. Conclusion

This project employs the concepts of relational database management including checking for functional dependencies, entity-relationships models, normalization etc. to build a database project for wedding planning management. All the entities involved in wedding planning have been identified and related to show dependencies and parent-child hierarchies. The entities have been further elaborated using their attributes which are populated at various levels in the web application.

The web application for wedding planning not only maintains the flow of control between pages to pages to abide by database constraints like null check, unique check, foreign key constraints but also maintains the consistency among transactions - a particular photographer booked for one particular wedding date, cannot be selected by other customer for same wedding date. Hence the ACID property of the database transactions has been maintained.