Cisc 332 Project - Part 3

Adam Perron, Brandon Bloch, Shannon Klett

Sections

Section 1: All Assumptions

Application Design

- There will be very few administrative users, and at least one will have direct access to the database. Therefore we do not need a front end feature for requesting, adding, or removing admin privileges. If there is a new admin user, they can be upgraded directly in the database.
- Suppliers should NOT be allowed to add reviews on their own properties.
- Suppliers should be allowed to make bookings on their own properties.
- Users can either search by all districts or one, all types or one, all features or one, maximum price or any combination of these.

Overall Database Design

- We chose to give lots of extra space for varchars.
    - We're assuming our database won't be big enough for this excess to matter.
    - Better safe than sorry!
- We're using tables for constant string values, such as booking status type, property type, faculty, and degree type.
    - This will help keep data consistent in our system, and allow us to easily add new types.
- NULL values are generally not allowed unless we have good reason. Columns containing strings will default to an empty string, which, unlike NULL, will not cause type errors when processing in frontend.
- Delete and cancel action actually remove the record from the database, rather than using a "deleted" column.

Bookings

- Bookings have their own id to make them easily searchable.

Rentals

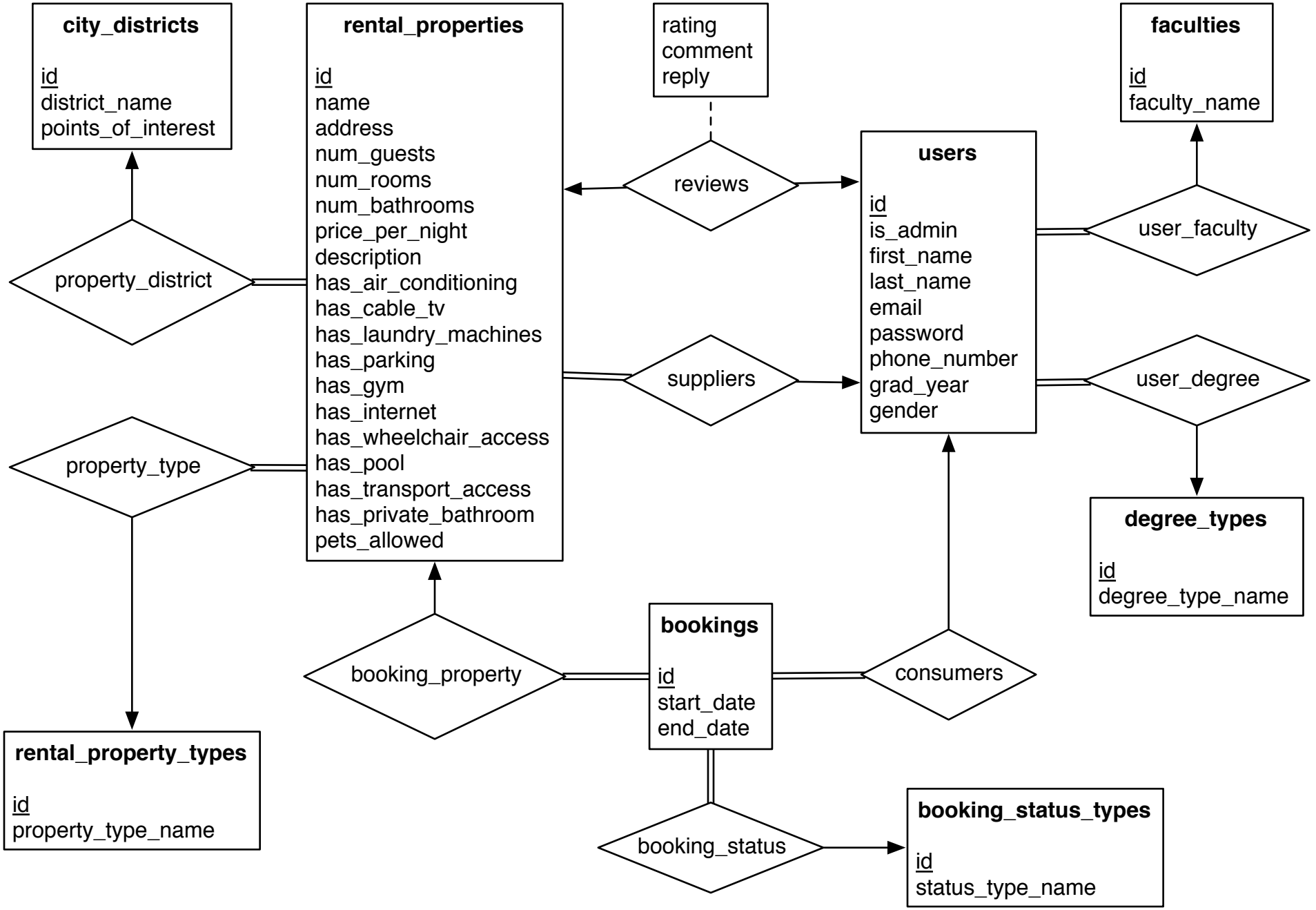
- All properties must be in exactly one district. Districts do not overlap. If the property is on the border of districts, supplier must select one.
    - The same applies to property type and supplier; every property must have exactly one type and supplier.
- price_per_night is in whole Canadian dollars (no cent values are allowed), and represents the price for one day/night (ie noon day 1 to 11am day 2). Total price for a booking will be this value multiplied by the number of days.
- Features are listed as individual boolean columns (which are actually type tinyint), to make them easily searchable. Columns can be easily added and the default value is false.

Reviews

- A user can leave at most one comment/rating on a property, and the supplier can only reply once.
- When giving a review, users must give a rating, but the comment is optional.

Users

- Longest phone number should be 15 digits, plus 6 digits for extension , and extra space just in case.
- grad_year is the 4 digit number representing the year they graduated (ie 2016). If user has multiple degrees from Queen's we'll use the most recent year.
- Gender is a varchar field to allow for non-binary genders. Users can identify as they wish.

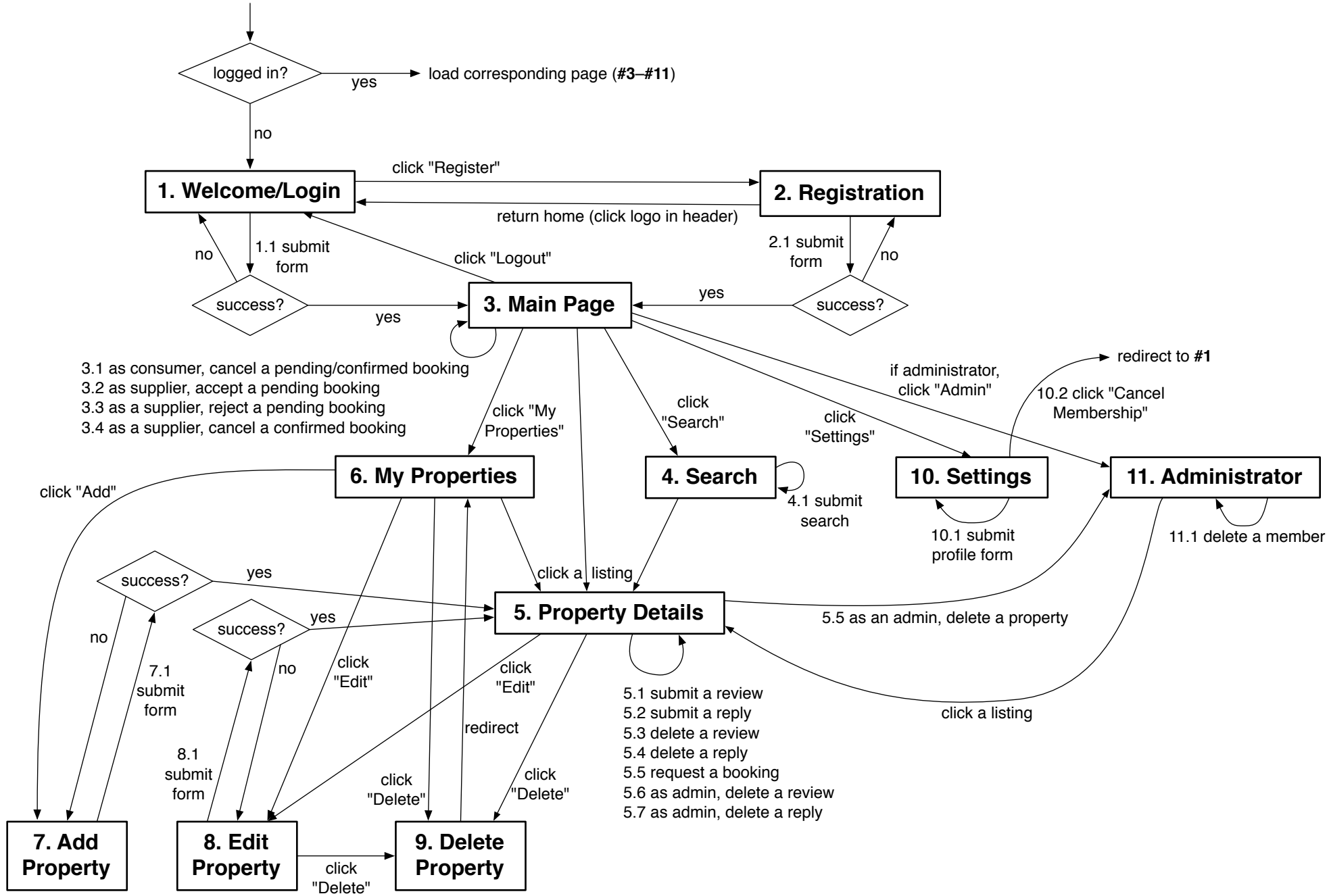

city_districts
id
district_name
points_of_interest
rental_properties
id
name
address
num_guests
num_rooms
num_bathrooms
price_per_night
description
has_air_conditioning
has_cable_tv
has_laundry_machines
has_parking
has_gym
has_internet
has_wheelchair_access
has_pool
has_transport_access
has_private_bathroom
pets_allowed
rating
comment
reply
faculties
id
faculty_name
reviews
users
id
is_admin
first_name
last_name
email
password
phone_number
grad_year
gender
user_faculty
property_district
suppliers
user_degree
property_type
degree_types
id
degree_type_name
booking_property
bookings
id
start_date
end_date
consumers
rental_property_types
id
property_type_name
booking_status
booking_status_types
id
status_type_name

Section 3: Relational Schema

```
CREATE TABLE `booking_status_types` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `status_type_name` varchar(50) NOT NULL DEFAULT '',
  PRIMARY KEY (`id`)
)

CREATE TABLE `bookings` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `consumer_id` int(11) unsigned NOT NULL,
  `property_id` int(11) unsigned NOT NULL,
  `start_date` date NOT NULL,
  `end_date` date NOT NULL,
  `status_id` int(2) unsigned NOT NULL,
  PRIMARY KEY (`id`),
  KEY `consumer_id` (`consumer_id`),
  KEY `property_id` (`property_id`),
  KEY `status_id` (`status_id`),
  CONSTRAINT `bookings_ibfk_1` FOREIGN KEY (`consumer_id`) REFERENCES `users` (`id`),
  CONSTRAINT `bookings_ibfk_2` FOREIGN KEY (`property_id`) REFERENCES `rental_properties` (`id`),
  CONSTRAINT `bookings_ibfk_3` FOREIGN KEY (`status_id`) REFERENCES `booking_status_types` (`id`)
)

CREATE TABLE `city_districts` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `district_name` varchar(255) NOT NULL DEFAULT '',
  `points_of_interest` text NOT NULL,
  PRIMARY KEY (`id`)
)

CREATE TABLE `degree_types` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `degree_type_name` varchar(50) NOT NULL DEFAULT '',
  PRIMARY KEY (`id`)
)

CREATE TABLE `faculties` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `faculty_name` varchar(50) NOT NULL DEFAULT '',
  PRIMARY KEY (`id`)
)

CREATE TABLE `rental_properties` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  `supplier_id` int(11) unsigned NOT NULL,
```

```
  `address` varchar(255) NOT NULL,
  `district_id` int(11) unsigned NOT NULL,
  `property_type_id` int(2) unsigned NOT NULL,
  `num_guests` int(10) unsigned NOT NULL DEFAULT '0',
  `num_rooms` int(10) unsigned NOT NULL DEFAULT '0',
  `num_bathrooms` int(10) unsigned NOT NULL,
  `price` int(5) NOT NULL,
  `description` text NOT NULL,
  `has_air_conditioning` tinyint(11) NOT NULL DEFAULT '0',
  `has_cable_tv` tinyint(11) NOT NULL DEFAULT '0',
  `has_laundry_machines` tinyint(11) NOT NULL DEFAULT '0',
  `has_parking` tinyint(11) NOT NULL DEFAULT '0',
  `has_gym` tinyint(11) NOT NULL DEFAULT '0',
  `has_internet` tinyint(11) NOT NULL DEFAULT '0',
  `pets_allowed` tinyint(11) NOT NULL DEFAULT '0',
  `has_wheelchair_access` tinyint(11) NOT NULL DEFAULT '0',
  `has_pool` tinyint(11) NOT NULL DEFAULT '0',
  `has_transport_access` tinyint(11) NOT NULL DEFAULT '0',
  `has_private_bathroom` tinyint(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`),
  KEY `district_id` (`district_id`),
  KEY `supplier_id` (`supplier_id`),
  KEY `property_type_id` (`property_type_id`),
  CONSTRAINT `rental_properties_ibfk_1` FOREIGN KEY (`district_id`) REFERENCES `city_districts` (`id`),
  CONSTRAINT `rental_properties_ibfk_2` FOREIGN KEY (`supplier_id`) REFERENCES `users` (`id`),
  CONSTRAINT `rental_properties_ibfk_3` FOREIGN KEY (`property_type_id`) REFERENCES
`rental_property_types` (`id`)
)

CREATE TABLE `rental_property_types` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `property_type_name` varchar(50) NOT NULL DEFAULT '',
  PRIMARY KEY (`id`)
)

CREATE TABLE `reviews` (
  `consumer_id` int(11) unsigned NOT NULL,
  `property_id` int(11) unsigned NOT NULL,
  `rating` int(1) unsigned NOT NULL,
  `comment` text NOT NULL,
  `reply` text NOT NULL,
  PRIMARY KEY (`consumer_id`,`property_id`),
  KEY `property_id` (`property_id`),
  CONSTRAINT `reviews_ibfk_1` FOREIGN KEY (`consumer_id`) REFERENCES `users` (`id`),
  CONSTRAINT `reviews_ibfk_2` FOREIGN KEY (`property_id`) REFERENCES `rental_properties` (`id`)
)
```

```
CREATE TABLE `users` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `is_admin` tinyint(1) NOT NULL DEFAULT '0',
  `first_name` varchar(50) NOT NULL,
  `last_name` varchar(50) NOT NULL,
  `email` varchar(100) NOT NULL,
  `password` varchar(255) NOT NULL,
  `phone_number` varchar(30) NOT NULL,
  `grad_year` int(4) NOT NULL,
  `faculty_id` int(2) unsigned NOT NULL,
  `degree_type_id` int(2) unsigned NOT NULL,
  `gender` varchar(255) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `faculty_id` (`faculty_id`),
  KEY `degree_type_id` (`degree_type_id`),
  CONSTRAINT `users_ibfk_1` FOREIGN KEY (`faculty_id`) REFERENCES `faculties` (`id`),
  CONSTRAINT `users_ibfk_2` FOREIGN KEY (`degree_type_id`) REFERENCES `degree_types` (`id`)
)
```


logged in?
yes
load corresponding page (#3–#11)
no
1. Welcome/Login
click "Register"
2. Registration
return home (click logo in header)
no
1.1 submit form
success?
yes
click "Logout"
3. Main Page
2.1 submit form
no
success?
yes
3.1 as consumer, cancel a pending/confirmed booking
3.2 as supplier, accept a pending booking
3.3 as a supplier, reject a pending booking
3.4 as a supplier, cancel a confirmed booking
click "My Properties"
click "Search"
click "Settings"
if administrator, click "Admin"
redirect to #1
10.2 click "Cancel Membership"
6. My Properties
4. Search
4.1 submit search
10. Settings
11. Administrator
10.1 submit profile form
11.1 delete a member
click "Add"
success?
yes
no
7.1 submit form
success?
yes
no
8.1 submit form
click a listing
5. Property Details
5.5 as an admin, delete a property
click "Edit"
click "Edit"
redirect
click "Delete"
click "Delete"
5.1 submit a review
5.2 submit a reply
5.3 delete a review
5.4 delete a reply
5.5 request a booking
5.6 as admin, delete a review
5.7 as admin, delete a reply
click a listing
7. Add Property
8. Edit Property
click "Delete"
9. Delete Property

Section 5: SQL Interactions in Each State + Output

Structure Overview:

0. Initialize
1. Login
    - Event 1.1: Submit Login Form
2. Registration
    - Event 2.1: Submit Registration Form
3. Home (My Bookings)
    - On Page Load
    - Event 3.1: Cancel Booking as Consumer
    - Event 3.2: Accept Booking as Supplier
    - Event 3.3: Reject Booking as Supplier
    - Event 3.4: Cancel Booking as Supplier
4. Search
    - Event 4.1: Submit Search
5. Property Details
    - On Page Load
    - Event 5.1: Submit Review
    - Event 5.2: Submit Reply
    - Event 5.3 and 5.6: Delete Review
    - Event 5.4 and 5.7: Delete Reply
    - Event 5.5: Submit Booking Request
6. My Properties
    - On Page Load
7. Add Property
    - Event 7.1: Submit form
8. Edit Property
    - Event 8.1: Submit form
9. Delete Property
    - On Page Load
10. User Settings
    - On Page Load
    - Event 10.1: Submit form
    - Event 10.2: Cancel Membership
11. Administrator
    - On Page Load
    - Event 11.1: Click “delete member”

0. Initialize

We decided to load our static tables (booking_status_types, city_districts, degree_types, rental_property_types) into PHP to reduce complex queries and to enable additional functionality like using the ID values in select lists and doing more robust match-checking using integers rather than strings. The first time

data from one of these static tables, we load the table in and set a variable. Future requests for this data will see that the variable is set and not call the database again.

Action: Get data from static tables.
SQL Queries:
SELECT id, district_name, points_of_interest FROM city_districts
SELECT id, property_type_name FROM rental_property_types
SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_SCHEMA = "cisc332project" AND TABLE_NAME = "rental_properties"
SELECT id, status_type_name FROM booking_status_types
SELECT id, degree_type_name FROM degree_types
SELECT id, faculty_name FROM faculties

SQL Output:
(rest cut off)

id	district_name
1	Bel-Air
2	Santa Monica
3	Beverly Crest
4	Westwood
5	Downtown
6	West Hollywood
7	Little Tokyo
8	Chinatown
9	Mid-City
10	Hollywood
11	LAX
12	Century City
13	Studio City
14	Brentwood
15	Silver Lake
16	Lincoln Heights
17	Beverly Hills
18	El Sereno

COLUMN_NAME
id
name
supplier_id
address
district_id
property_type_id
num_guests
num_rooms
num_bathrooms
price
description
has_air_conditioning
has_cable_tv
has_laundry_machines
has_parking
has_gym
has_internet
pets_allowed
has_wheelchair_access
has_pool
has_transport_access
has_private_bathroom

id	property_type_name
1	Entire Apt/House
2	Private Room
3	Shared Room

id	status_type_name
1	requested
2	confirmed
3	rejected

id	degree_type_name
1	BA
2	BCMP
3	BFA
4	BMUS
5	BPHE
6	BSC
7	BED
8	BENG
9	BCOMM
10	MBA
11	MD
12	MA
13	MES
14	LLM
15	PHD
16	OTHER

1. Login

Event 1.1: Submit Login Form

Action: Check if login valid. We will use a PHP function to hash the password string in order to keep it secure.
SQL Query:
SELECT id, password FROM users WHERE email = "davidthompson@teleworm.us"

Sample Output:

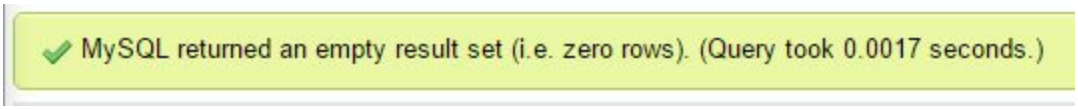
id	password
4	$2y$12$cWUF6kf2F5kOKcAcVxrQEuyg0Kuzh5qq4jETLOBLq7Y...

2. Registration

Event 2.1: Submit Registration Form

Action: Check that email address is unique in database. The SELECT 1 will return 1 if the email exists in the database and nothing if it doesn't. It requires less processing from the databases and transfers only the necessary information.
SQL Query:
SELECT 1 FROM users WHERE email = "12sak2@queensu.ca"

SQL Output:



Action: If email is unique, add user to database.
SQL Query:
INSERT INTO users (id, is_admin, first_name, last_name, email, password, phone_number, grad_year, faculty_id, degree_type_id, gender) VALUES
(NULL, "0", "Shannon", "Klett", "12sak2@queensu.ca", "$2y$12$wwCrm0V8hc1bEvzGbfVAKugSlrRwxNW03hnEIo0HE/SNtFsjMp1cC", "+1 902-111-1101", "2016", "1", "2", "Female")

Sample Output:

id	is_admin	first_name	last_name	email	password	phone_number	grad_year	faculty_id	degree_type_id	gender
27	0	Shannon	Klett	12sak2@queensu.ca	$2y$12$wwCrm0V8hc1bEvzGbfVAKugSlrRwxNW03hnEIo0HE/S...	+1 902-111-1101	2016	1	2	Female

3. Home (My Bookings)

On Page Load

Action: Get all bookings requested by user.
SQL Query:
SELECT * FROM bookings WHERE consumer_id = 3 ORDER BY start_date ASC
Sample Output:

id	consumer_id	property_id	start_date	end_date	status_id
1	3	1	2016-03-14	2016-03-21	1
2	3	13	2016-06-01	2016-06-08	1

Action: Get all bookings on a user's accommodations.
SQL Query:
SELECT * FROM bookings b JOIN rental_properties p WHERE b.property_id = p.id AND p.supplier_id = 1 ORDER BY b.start_date ASC

Sample Output:

id	consumer_id	property_id	start_date	end_date	status_id	id	name	supplier_id	address	district_id	pr
10	27	1	2016-04-01	2016-04-13	1	1	Signature Beverly Hills Studio	1	707 N Beverly Dr, Beverly Hills, CA...	17	

Event 3.1: Cancel Booking as Consumer

Action: Get supplier email in order to notify them, and information about booking so email is informative.
SQL Query:
SELECT supplier.first_name as supplier_first, supplier.last_name as supplier_last, supplier.email as supplier_email, consumer.first_name as consumer_first, consumer.last_name as consumer_last, rental_properties.name as property_name, start_date, end_date, status_type_name
FROM rental_properties JOIN bookings ON property_id = rental_properties.id
JOIN users supplier ON supplier_id = supplier.id
JOIN users consumer ON consumer_id = consumer.id
JOIN booking_status_types ON status_id = booking_status_types.id
WHERE bookings.id = 4

Sample Output:

supplier_first	supplier_last	supplier_email	consumer_first	consumer_last	property_name	start_date	end_date	status_type_name
Jodi	Navarro	jodi_t@rogers.ca	Gerald	Coppola	Private Room in Venice Beach	2016-11-03	2016-11-10	confirmed

Action: Remove booking from database
SQL Query:
DELETE FROM bookings
WHERE bookings.id = 4

Sample Output:

Before -

id	consumer_id	property_id	start_date	end_date	status_id
1	3	1	2016-03-14	2016-03-21	1
2	3	13	2016-06-01	2016-06-08	1
3	4	2	2016-07-11	2016-07-18	2
4	5	4	2016-11-03	2016-11-10	2
5	8	4	2016-04-20	2016-04-13	3

After -

id	consumer_id	property_id	start_date	end_date	status_id
1	3	1	2016-03-14	2016-03-21	1
2	3	13	2016-06-01	2016-06-08	1
3	4	2	2016-07-11	2016-07-18	2
5	8	4	2016-04-20	2016-04-13	3

Event 3.2: Accept Booking as Supplier

Action: Status of booking updated in database
SQL Query:
UPDATE bookings SET status_id = 2 WHERE id = 1

Sample Output:

Before -

id	consumer_id	property_id	start_date	end_date	status_id
1	3	1	2016-03-14	2016-03-21	1
2	3	13	2016-06-01	2016-06-08	1
3	4	2	2016-07-11	2016-07-18	2
4	5	4	2016-11-03	2016-11-10	2
5	8	4	2016-04-20	2016-04-13	3

After -

id	consumer_id	property_id	start_date	end_date	status_id
1	3	1	2016-03-14	2016-03-21	2
2	3	13	2016-06-01	2016-06-08	1
3	4	2	2016-07-11	2016-07-18	2
4	5	4	2016-11-03	2016-11-10	2
5	8	4	2016-04-20	2016-04-13	3

Action: Get information to notify consumer
SQL Query:
SELECT first_name, last_name, email, rental_properties.name as property_name, start_date, end_date
FROM users JOIN bookings ON consumer_id = users.id
JOIN rental_properties ON property_id = rental_properties.id
WHERE bookings.id = 1

Sample Output:

first_name	last_name	email	property_name	start_date	end_date
Eva	Yarbrough	eva_yar_89@gmail.com	Signature Beverly Hills Studio	2016-03-14	2016-03-21

Event 3.3: Reject Booking as Supplier

Action: Status of booking updated in database (status must be pending)
SQL Query:
UPDATE bookings SET status_id = 3 WHERE id = 1

Sample Output:

Before -

id	consumer_id	property_id	start_date	end_date	status_id
1	3	1	2016-03-14	2016-03-21	1
2	3	13	2016-06-01	2016-06-08	1
3	4	2	2016-07-11	2016-07-18	2
4	5	4	2016-11-03	2016-11-10	2
5	8	4	2016-04-20	2016-04-13	3

After -

id	consumer_id	property_id	start_date	end_date	status_id
1	3	1	2016-03-14	2016-03-21	3
2	3	13	2016-06-01	2016-06-08	1
3	4	2	2016-07-11	2016-07-18	2
4	5	4	2016-11-03	2016-11-10	2
5	8	4	2016-04-20	2016-04-13	3

Action: Get information to notify consumer (same SQL query as Event 3.2: Accept Booking as Supplier)

Event 3.4: Cancel Booking as Supplier

Action: Booking removed from database and customer notified
SQL Query:
DELETE FROM bookings WHERE id = 3
Sample Output:

id	consumer_id	property_id	start_date	end_date	status_id
1	3	1	2016-03-14	2016-03-21	1
2	3	13	2016-06-01	2016-06-08	1
4	5	4	2016-11-03	2016-11-10	2
5	8	4	2016-04-20	2016-04-13	3

Action: Get information to notify consumer (same SQL query as Event 3.2: Accept Booking as Supplier)

4. Search

Event 4.1: Submit Search

Action: Get list of properties that match search parameters. Users can search by any combination of: district, type, features, price. Different queries are used based on the combination of parameters searched; one example is shown here, where a specific district and property type were chosen, and price was capped at $300.
SQL Query:

SELECT * FROM rental_properties WHERE property_type_id = 2 AND price <= 300 AND district_id = 6
ORDER BY name ASC
SQL Output:

id	name	supplier_id	address	district_id	property_type_id	num_guests	num_rooms	num_bathrooms	price	description	has_air_conditioning	has_cable_tv	has_laundry_machines	has
13	Guest Suite-Heart of West Hollywood	6	8540 Melrose Ave, West Hollywood, CA 90069, USA	6	2	2	1	1	204	Large guest suite w/queen size "heavenly" pillowto...	0	0	0	

5. Property Details

On Page Load

Action: Get property details
SQL Query:
SELECT * FROM rental_properties WHERE id = 1

Sample Output:
(The rest is cut off)

id	name	supplier_id	address	district_id	property_type_id	num_guests	num_rooms	num_bathrooms	price	description	has_air_conditioning	has_cable_tv	has_laund
1	Signature Beverly Hills Studio	1	707 N Beverly Dr, Beverly Hills, CA 90210, USA	17	1	2	1	1	109	Located in exclusive Beverly Hills, just steps fro...	1	1	

Action: Get reviews on the property
SQL Query:
SELECT * FROM reviews WHERE property_id = 1

Sample Output:

consumer_id	property_id	rating	comment	reply
2	1	4	What a lovely place. Truly a paradise! Would definitely love to come back wh...	Thanks!
12	1	3	Great place to stay advert welcoming hosts	

Action: Check if user has previously made a booking on property (then display 'add review' and delete review buttons)
SQL Query:
SELECT 1 as num_bookings
FROM rental_properties JOIN bookings ON property_id = rental_properties.id
JOIN users on supplier_id = users.id
WHERE rental_properties.id = 1 AND consumer_id = 3 AND consumer_id != supplier_id

Sample Output:

num_bookings
1

Action: Check if user is the supplier (then show reply and delete reply buttons)

SQL Query:
SELECT COUNT(*)
FROM rental_properties
WHERE id = 1 AND supplier_id = 1

Sample Output:

COUNT(*)
1

Event 5.1: Submit Review

Action: Add review to database.
SQL Query:
INSERT INTO reviews (consumer_id, property_id, rating, comment) VALUES (8, 4, 3, "It was alright")

Sample Output:

consumer_id	property_id	rating	comment	reply
8	4	3	It was alright	

Event 5.2: Submit Reply

Action: Add reply to database.
SQL Query:
UPDATE reviews
SET reply = "How could it improve?"
WHERE consumer_id = 8 and property_id = 5

Sample Output:

consumer_id	property_id	rating	comment	reply
8	5	3	It was alright	How could it improve?

Event 5.3 and 5.6: Delete Review

Action: Delete review from database.
SQL Query:
DELETE FROM reviews
WHERE consumer_id = 19 and property_id = 14

Sample Output:
Before -

consumer_id	property_id	rating	comment	reply
6	9	4	Nice host at nice location. Quiet street for sleep...	
12	3	4	Perfect location close to Sunset Blvd, absolutely ...	Thank you for your feedback!
19	14	5	Easy to find in a quiet area that is also close to...	

After -

consumer_id	property_id	rating	comment	reply
6	9	4	Nice host at nice location. Quiet street for sleep...	
12	3	4	Perfect location close to Sunset Blvd, absolutely ...	Thank you for your feedback!

Event 5.4 and 5.7: Delete Reply

Action: Delete reply from review in database.
SQL Query:
UPDATE reviews
SET reply = ""
WHERE consumer_id = 6 and property_id = 9

Sample Output:
Before -

consumer_id	property_id	rating	comment	reply
6	9	4	Nice host at nice location. Quiet street for sleep...	Thank you!

After -

consumer_id	property_id	rating	comment	reply
6	9	4	Nice host at nice location. Quiet street for sleep...	

Sample Output:
(2 is gone)

id	name	supplier_id	address	district_id	property_type_id	num_guests	num_rooms	num_bathrooms	price	description	has_air_conditioning	ha
1	Signature Beverly Hills Studio	1	707 N Beverly Dr, Beverly Hills, CA 90210, USA	17	1	2	1	1	109	Located in exclusive Beverly Hills, just steps fro...	1	
3	STAY IN SILVER LAKE!	3	1699 Rotary Dr, Los Angeles, CA 90026, USA	15	2	2	1	1	85	Your private bedroom with a view. Be greeted by my...	0	
4	Private Room in Venice Beach	7	1708 Linden Ave, Venice, CA 90291, USA	5	2	3	1	1	92	See the ocean from the window of this private room...	1	
5	HOLLYWOOD HIGHLAND - MODERN 1 BED	18	1315 N Kingsley Dr, Los Angeles, CA 90027, USA	10	1	3	1	1	140	Walk to the Bowl, Walk of Fame, Kodak Theater. Enj...	0	
6	Hollywood Holiday (Private Suite)	8	446 N Hobart Blvd, Los Angeles	10	1	2	1	2	170	Welcome to the middle of Hollywood	1	

Event 5.5: Submit Booking Request

Action: Update database with new booking request.
SQL Query:
INSERT INTO bookings (consumer_id, property_id, start_date, end_date, status_id)
VALUES (12, 1, "2016-04-01", "2016-04-10", 1)

Sample Output:
Before-

id	consumer_id	property_id	start_date	end_date	status_id
1	3	1	2016-03-14	2016-03-21	3
2	3	13	2016-06-01	2016-06-08	1
3	4	2	2016-07-11	2016-07-18	2
4	5	4	2016-11-03	2016-11-10	2
5	8	4	2016-04-20	2016-04-13	3

After-

id	consumer_id	property_id	start_date	end_date	status_id
1	3	1	2016-03-14	2016-03-21	3
2	3	13	2016-06-01	2016-06-08	1
3	4	2	2016-07-11	2016-07-18	2
4	5	4	2016-11-03	2016-11-10	2
5	8	4	2016-04-20	2016-04-13	3
6	12	1	2016-04-01	2016-04-10	1

Action: Get information to notify supplier of request.
SQL Query:
SELECT supplier.first_name as supplier_first, supplier.last_name as supplier_last, supplier.email as supplier_email,
consumer.first_name as consumer_first, consumer.last_name as consumer_last,
rental_properties.name as property_name, start_date, end_date
FROM rental_properties JOIN bookings ON property_id = rental_properties.id
JOIN users supplier ON supplier_id = supplier.id
JOIN users consumer ON consumer_id = consumer.id
WHERE bookings.id = 1

Sample Output:

supplier_first	supplier_last	supplier_email	consumer_first	consumer_last	property_name	start_date	end_date
Steven	Henley	steven_soccer_4ever@hotmail.com	Eva	Yarbrough	Signature Beverly Hills Studio	2016-03-14	2016-03-21

6. My Properties

User can see a list of their own properties, as well as buttons for adding a property, editing a property, and deleting one. These buttons will open a popup. We will pass information from the My Properties page to these popups, so they don't require their own sql queries for page load.

On Page Load

Action: Get all properties and details for current user.
SQL Query:
SELECT * FROM `rental_properties` WHERE supplier_id = 1 ORDER BY name ASC

SQL Output:
(rest cut off)

id	name	supplier_id	address	district_id	property_type_id	num_guests	num_rooms	num_bathrooms	price	description
9	Nice, quite place in lovely house	10	529 E Temple St, Los Angeles, CA 90012, USA	7	3	1	1	1	32	This is clean and comfy bunk bed place in a house,...
11	Guest Cottage Near the Sea!	10	14919 La Cumbre Dr, Pacific Palisades, CA 90272, U...	19	1	2	1	1	156	Our lovely, comfortable guest cottage is just bloc...

7. Add Property

Event 7.1: Submit form

Action: Add new (valid) property to database.
SQL Query:
INSERT INTO rental_properties (name, supplier_id, address, district_id, property_type_id, num_guests, num_rooms, num_bathrooms, price, description, has_air_conditioning, has_cable_tv, has_laundry_machines, has_parking, has_gym, has_internet, pets_allowed, has_wheelchair_access, has_pool, has_transport_access, has_private_bathroom)
VALUES ("NEW", 1, "101 Main St, Beverly Hills, CA, 90210, USA", 17, 2, 1, 1, 1, 99, "You'll love it", 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0)

Sample Output:
(Rest cut off)

id	name	supplier_id	address	district_id	property_type_id	num_guests	num_rooms	num_bathrooms	price	description
16	NEW	1	101 Main St, Berverly Hills, CA, 90210, USA	17	2	1	1	1	99	You'll love it

8. Edit Property

Suppliers can see and edit their property information. Administrators can also edit property information.

Event 8.1: Submit form

Action: Update database with modifications for property. Users cannot change the property id.
SQL Query:

```
UPDATE rental_properties
SET
name = "NEW Signature Beverly Hills Studio",
property_type_id = 2,
num_guests = 1,
price = 120,
has_air_conditioning = 0,
has_cable_tv = 0
WHERE id = 1
```

Sample Output:

Before -

id	name	supplier_id	address	district_id	property_type_id	num_guests	num_rooms	num_bathrooms	price	description	has_air_conditioning	has_cable_tv	has_laun
1	Signature Beverly Hills Studio	1	707 N Beverly Dr, Beverly Hills, CA 90210, USA	17	1	2	1	1	109	Located in exclusive Beverly Hills, just steps fro...	1	1	

After -

id	name	supplier_id	address	district_id	property_type_id	num_guests	num_rooms	num_bathrooms	price	description	has_air_conditioning	has_cable_tv	has_laundr
1	NEW Signature Beverly Hills Studio	1	707 N Beverly Dr, Beverly Hills, CA 90210, USA	17	2	1	1	1	120	Located in exclusive Beverly Hills, just steps fro...	0	0	

9. Delete Property

This page will run when a user clicks “delete”. The delete button is only visible if the current user is the supplier or an admin, and the page redirects before deleting if they should not have access to it.

On Page Load

Action: Get information to notify any consumers with pending or confirmed bookings. If admin deletes property, also notify the supplier.
SQL Query:

```
SELECT bookings.id as booking_id, rental_properties.name as property_name, start_date, end_date,
status_type_name,
consumers.first_name as consumer_first, consumers.last_name as consumer_last, consumers.email as
consumer_email,
suppliers.first_name as supplier_first, suppliers.last_name as supplier_last, suppliers.email as supplier_email
FROM bookings JOIN rental_properties ON property_id = rental_properties.id
JOIN booking_status_types ON status_id = booking_status_types.id
JOIN users consumers ON consumer_id = consumers.id
JOIN users suppliers ON supplier_id = suppliers.id
```

WHERE (status_id = 1 OR status_id =2) AND bookings.id IN (SELECT bookings.id FROM bookings WHERE property_id = 4)

SQL Output:

booking_id	property_name	start_date	end_date	status_type_name	consumer_first	consumer_last	consumer_email	supplier_first	supplier_last	supplier_email
4	Private Room in Venice Beach	2016-11-03	2016-11-10	confirmed	Gerald	Coppola	geraldy_95@live.ca	Jodi	Navarro	jodi_t@rogers.ca

Action: As the owner of the property or an admin, delete the property and all dependent data from database. Queries will be executed in a transaction.
SQL Queries:
DELETE FROM reviews WHERE property_id = 16
DELETE FROM bookings WHERE property_id = 16
DELETE FROM rental_properties WHERE id = 16

Sample Output:
(16 is gone)

13	Guest Suite-Heart of West Hollywood	6	8540 Melrose Ave, West Hollywood, CA 90069, USA	6	2	2	1	1	204	Large g suite w size "heave pillowtc
14	Private Room with Private Entrance	9	13044 Discovery Creek, Los Angeles, CA 90094, USA	20	2	2	1	1	152	We hav beautif private with a p fu...
15	Zen Paradise in Beverly Hills	4	9749 Sunset Blvd, Beverly Hills, CA 90210, USA	17	1	2	1	2	270	The fer energy balance museu quali...

elected: Edit Copy Delete Export

s: All Filter rows: Search this table

10. User Settings

On this page, users can see and edit all of the information in the users table, with the exceptions that id and is_admin cannot be edited by a user, and password can only be updated, but not seen in the page. Users can also cancel their membership to the site, removing all their bookings, reviews, replies, and properties.

On Page Load

Action: Get user information
SQL Query:
SELECT is_admin, first_name, last_name, phone_number, grad_year, faculty_name, degree_type_name, gender
FROM users JOIN faculties ON faculty_id = faculties.id
JOIN degree_types ON degree_type_id = degree_types.id
WHERE users.id = 1

Sample Output:

is_admin	first_name	last_name	email	password	phone_number	grad_year	faculty_name	degree_type_name	gender
0	Steven	Henley	steven_soccer_4ever@hotmail.com	mjfJc9RS45bEPXYZT5kL	+1 905-952-3917	2010	Faculty of Arts and Science	BA	Male

Event 10.1: Submit form

Action: Update users table with new information. Id of modified row must match that of the current user.
SQL Query:

```
UPDATE users
SET
first_name = "S",
password = "Hj5545lmnk09wWzx51r",
gender = "Prefer not to disclose"
WHERE id = 1
```

Sample Output:

Before -

id	is_admin	first_name	last_name	email	password	phone_number	grad_year	faculty_id	degree_type_id	gender
1	0	Steven	Henley	steven_soccer_4ever@hotmail.com	mjfJc9RS45bEPXYZT5kL	+1 905-952-3917	2010	1	1	Male

After -

id	is_admin	first_name	last_name	email	password	phone_number	grad_year	faculty_id	degree_type_id	gender
1	0	S	Henley	steven_soccer_4ever@hotmail.com	Hj5545lmnk09wWzx51r	+1 905-952-3917	2010	1	1	Prefer not to disclose

Event 10.2: Cancel Membership

Action: Remove user from database, all their bookings, reviews and replies, and properties. These queries will be executed in a transaction.
SQL Queries:

```
DELETE r FROM reviews r JOIN rental_properties p ON r.property_id = p.id
WHERE p.id = 23
DELETE b FROM bookings b JOIN rental_properties p ON b.property_id = p.id
WHERE p.id = 23
DELETE FROM bookings WHERE consumer_id = 23
DELETE FROM rental_properties WHERE supplier_id = 23
DELETE FROM users WHERE id = 23
```

Sample Output:
(23 is gone)

20	0	Alma	Moretti	alma_moretti@live.com	28EENcwyQLuQASteH9ux	+39 0370 4698148	1996	4	11	Female
21	0	Blake	Flores	blake_flores@live.ca	eVJxQYpbby37BVLP62Qq	+1 418-818-6321	2006	1	1	Male
22	0	Estelle	St-Jean	estellexoxo123@hotmail.ca	fsMUqmtMfNkERg3j7rrA	+1 250-365-4685	2003	1	15	Female
24	0	Alice	Whitehead	maple_leafs_lover_32@yahoo.ca	Pm49YNw3XQK2M6MXdqVm	+1 705-624-1911	1994	1	1	Female
25	0	Mark	Powell	markhpowell@bell.ca	ZzbSWUDzp42dAPGX993D	+1 306-230-0157	2001	1	1	Male

11. Administrator

This page will has four tab views, System Users, Property Summary, Supplier Summary, and Consumer Summary. All the data for all four views is retrieved when the page is loaded. System Users shows a list of members and a delete button for each. Property Summary shows a list of all properties, with a summary of the total number of bookings and the average rating for the property. The Supplier Summary lists the total number

of bookings and average rating for all the properties listed by a supplier. Consumer Summary summarizes requested, confirmed, and rejected bookings by user. Most of this data is used by multiple tabs and for multiple different calculations, so much of it is retrieved without joining tables or using aggregate functions to enable re-use of the same original tuple data throughout the application rather than re-querying.

On Page Load

Action: Get all members.
SQL Query:
SELECT id, first_name, last_name, email FROM `users` ORDER BY last_name ASC

Sample Output:
(the rest is cut off)

id	first_name	last_name	email
14	Aladdin	Almasi	aladdinazimalmasi@hotmail.co.uk
10	Minnie	Bookman	minnie_crazy_fish_summer_z83@hotmail.com
12	Chow	Chu	chowchu@gmail.com
5	Gerald	Coppola	geraldy_95@live.ca
27	Shay	Dawg	sk@q.ca
21	Blake	Flores	blake_flores@live.ca
9	Terry	Foote	terryb_foote@hotmail.co.uk
1	Steven	Henley	steven_soccer_4ever@hotmail.com
20	Alma	Moretti	alma_moretti@live.com
7	Jodi	Navarro	jodi_t@rogers.ca
18	Doàn	Nhân	doannhan@hotmail.com
19	Lilian	Nordström	liliannordstrom@gmail.com
6	Frank	Norman	frank_norman@norman.ca
16	Karen	Park	kpark@outlook.ca
8	John	Parker	johnbparker@gmail.com
26	Adam	Perron	me@adamperron.com
15	Théodore	Pouchard	theodore_pouchard@yahoo.com

Action: Get all bookings
SQL Query:
SELECT id, consumer_id, property_id, status_id FROM bookings ORDER BY start_date ASC

Sample Output:

id	consumer_id	property_id	status_id
10	27	1	1
15	2	17	2
6	27	8	1
13	27	17	1
5	8	4	3
14	27	17	3
9	27	4	1
4	5	4	2

Action: Get all properties.
SQL Query:
SELECT id, name, supplier_id FROM rental_properties ORDER BY name ASC

Sample Output:

id	name	supplier_id
17	Adam's Place!	26
8	Cozy living room by the beach	13
10	Downtown Loft	2
11	Guest Cottage Near the Sea!	10
13	Guest Suite-Heart of West Hollywood	6
5	HOLLYWOOD - MODERN 1 BED	18
12	Hollywood Hills Lux A-List PoolView	5
9	Nice, quite place in lovely house	10
4	Private Room in Venice Beach	7
1	Signature Beverly Hills Studio	1
16	The Palace	27
7	Wonderful room in SM mountain view	12
15	Zen Paradise in Beverly Hills	4

Action: Get all review ratings.
SQL Query:
SELECT consumer_id, property_id, rating FROM reviews

Sample Output:

consumer_id	property_id	rating
2	1	4
6	9	4
12	1	3
27	4	3
27	17	5

Event 11.1: Click "delete member"

Runs the same set of queries as Event 10.2: Cancel Membership.

Section 6: Discussion

Problems Encountered + Solutions

One problem we encountered was that we had to re-write a lot of our queries in the final phase because we hadn't thought about user experience at the beginning. We modified a lot of our queries to return more useful information, or appear in a more useful way (ie sorted).

As well, we had troubles with the queries for the Admin page. We weren't really sure what sort of summary information would be useful, or the optimal query to get this information. We ended up doing separate queries for each field in our table for each user because that worked best with our object architecture (see next section).

Design and Implementation Decisions

Php Objects

A major decision made while implementing the business logic of our application to use an object-oriented approach. Tuples in the database are represented by corresponding objects in the application, encapsulating all database queries within the object and reducing the potential for issues with data vulnerability and integrity.

Objects allow for writing multiple different constructors and factory methods (eg. get a user record by ID or email address, or retrieve all the reviews on a single property) as well as encapsulation of getters, setters, and database access methods. Objects are either retrieved from the database or constructed; an insert method allows insertion of a newly-constructed object, an update method applies the new values of the object passed to the setters, and a delete method deletes the corresponding tuple from the database.

This change in organization of our application layer allows for better data protection and greater flexibility in the future, but involved changing a few of our original queries to make them more generic, allowing them to be used in multiple use cases. These changes are discussed in detail in **SQL Interaction in Each State + Output**.

SELECT 1

We have queries that check the database to see if something exists, such as if a user is already registered with a given email. The SELECT 1 statement will return 1 if the data exists in the database and nothing if it doesn't. It requires less processing from the databases and transfers only the necessary information.

Static Tables Loaded into PHP

In our database, we have four 'static tables' that act kind of like enums: booking_status_types, city_districts, degree_types, and rental_property_types. These tables map a string value to an int id number. This structure allows us to easily edit the strings values of these properties, as well as add new

ones. The change is contained within one table so it has minimal impact on our system. We decided to load our static tables into PHP to reduce extra complex queries. The first time we need data from one of these static tables, we load the table in and set a variable. Future requests for this data will see that the variable is set and not call the database again.

Technologies

For our frontend web application, we used HTML, CSS, JavaScript, and the Bootstrap framework. We also used Adobe Illustrator to pre-conceptualize our designs. We really wanted to ensure our layout was simple and intuitive for users.

Our backend is coded in PHP. We used Sequel Pro for SQL administration. It's a native app for SQL administration on Macs. You can access all the databases located on the local host and the web and manually edit the tables and their contents. It's simpler, faster, easier than PhpMyAdmin.

For our version control, we used Git and hosted our project on GitHub.

We chose these technologies because we all have experience with them, so they were easy to set up and work in. Overall we had a good experience with them and would choose them again for future projects.

What We Would Do Differently

Next time we have a similar task, we should think about the users earlier. In the beginning, we forgot some pretty major features for users, such as including a password field in our database. In general, we often thought about the technical details, such as how to store and query the data, but not about how users will interact with the system. Even when designing user flow, our main focus was just displaying our data and not really what the user would want in our system. This oversight led to a sub-optimal user interface. It's usable, but there are some missing features that would be useful for users.

Possible Improvements and Extensions

- Admin
    - Ability for user to request admin privileges, an admin to see these requests and grant or remove them, or to remove admin privileges.
    - Search for user by name or email and see a summary of their information.
    - Search for a property by name and see a summary of its information.
    - Interface for adding/updating/removing data from our static tables: districts, etc (static tables)
- General
    - Allow users to undo deletes or retrieve deleted data by adding a "deleted" column to our bookings, rental_properties, reviews, and users tables, rather than permanently removing rows from our database

- Allowing users to search by multiple districts, types, and features at a time.
- Search for property by name
- Search for property by suppliers
- Member profile pages where you can see their properties and some basic personal info.
- Listing check-in and check-out times on a property

Section 7: User Guide

System Setup

To setup our QBNB system, the administrator should clone our git repository located at https://github.com/shannonklett/qbnb. From there, they can add the code to their server. To set up the database, we have included a .sql file that can be imported through phpmyadmin. This file includes the DDL for the database schema, as well as the static content, such as degree types and booking status types. We have also included city districts, but the administrator can change those if they would like to use our system outside of LA. The user should register through the site (to ensure their password is properly hashed), and then go into the database and change their is_admin field to 1 (true). They can repeat this process for any other administrators.

All Members

1) Register as a new member.

On the homescreen, select the “Register” button. Enter your information on the Registration page and click the “Register” button.

2) Login to system

On the homescreen, enter email and password you used to sign up for our system then click the “Login” button.

3) Update member profile.

Select the “SETTINGS” Page from the menu in the top right. Modify any information you would like and click the “Save Changes” button.

4) Cancel their membership.

Select the “SETTINGS” Page from the menu in the top right. Click the “Cancel Membership” button.

As a supplier

1) View all accommodations.

Select the “MY PROPERTIES” Page from the menu in the top right. You will see your properties listed on this page.

2) Add a new accommodation.

Select the “MY PROPERTIES” Page from the menu in the top right. Click the “Add Property” button. Fill in the information and click the “Submit” button.

3) Remove an existing accommodation.

Select the “MY PROPERTIES” Page from the menu in the top right. Click the “X” button in the bottom right corner of a property.

4) Update an existing accommodation.

Select the “MY PROPERTIES” Page from the menu in the top right. Click the pencil icon in the bottom right corner of a property. Fill in the information and click the “Submit” button.

5) Respond to booking request.

Navigate to the home page by clicking the logo in the top left corner of the page. Select either check mark or the “X” on the booking request depending on your preferred response.

6) Reply to a comment on one of their accommodations.

Navigate to the property details page for the accommodation, either through the “MY PROPERTIES” page or through the “SEARCH” page. Find the comment you wish to reply to. Click the “Reply” button, type your reply, and click the “Submit” button.

As a consumer

1) Search accommodations by district, type, features, price.

Select the “SEARCH” Page from the menu in the top right. Enter your search terms using the drop down menus. You may search by any combination of the fields (district, type, features, price), leaving the other fields empty. Click the “Search” button to see matching properties.

2) List all the ratings and comments for an accommodation.

Navigate to the property details page for the accommodation, either through the home page or through the “SEARCH” page. Scroll down until you see the ratings and comments.

3) List availability for an accommodation.

Navigate to the property details page for the accommodation, either through the home page or through the “SEARCH” page. Scroll down until you see the list of current bookings.

4) Request a booking.

Navigate to the property details page for the accommodation, either through the home page or through the “SEARCH” page. Enter your desired arrival and departure dates then click the “Book” button.

5) List all a consumer’s bookings.

Navigate to the home page by clicking the logo in the top left corner of the page. You will see a list of all your current bookings.

6) Add a comment/rating for one of their booked accommodations.

Navigate to the property details page for the accommodation, either through the home page or through the “SEARCH” page. Write a review (optional) and select a rating (required) then click the “Submit” button.

7) Cancel a booking.

Navigate to the home page by clicking the logo in the top left corner of the page. Click the “X” next to the booking.

Administrators

1) Delete a member and all their accommodations.

Select the “ADMIN” Page from the menu in the top right. Select the “Users” tab. Click the “X” button beside the desired user.

2) Delete an accommodation.

Navigate to the property details page for the accommodation, either through the “MY PROPERTIES” page, the “SEARCH” page, or the “ADMIN” page. Click the red “X” button below the photo of the property.

3) Summarize bookings and ratings per accommodation.

Select the “ADMIN” Page from the menu in the top right. Select the “Property Summary” tab.

4) Summarize bookings and ratings per supplier.

Select the “ADMIN” Page from the menu in the top right. Select the “Supplier Summary” tab.

5) Summarize booking activity per consumer.

Select the “ADMIN” Page from the menu in the top right. Select the “Consumer Summary” tab.