

CISC 332

QBnB Project Deliverable 3

Zachary Baum 10090150

Vinyas Harish 10089169

March 31st, 2016

Table of Contents

CONTINUED ASSUMPTIONS	1
ENTITY RELATIONSHIP DIAGRAM	3
RELATIONAL SCHEMA	4
TABLE DUMPS	6
SQL STATEMENTS & SAMPLE OUTPUTS	9
GENERIC STATEMENTS - BY USER	9
MEMBER	9
OWNER	9
CONSUMER	10
ADMIN	11
DEMO STATEMENTS - BY USER	12
MEMBER	12
OWNER	13
CONSUMER	14
ADMIN	17
STATE MACHINE DIAGRAM	20
DISCUSSION	21
PROJECT OVERVIEW	21
CORE FUNCTIONALITY ACHIEVED	21
ALL MEMBERS:	21
OWNERS:	21
CONSUMERS:	21
ADMINISTRATORS:	21
DESIGN AND IMPLEMENTATION DECISIONS	22
DESIGN	22
IMPLEMENTATION	24
TECHNOLOGIES AND TOOLS USED	24
LANGUAGES	24
TECHNOLOGIES & FRAMEWORKS USED:	25
PROBLEMS FACED	28
CRITICAL REFLECTION AND FUTURE DEVELOPMENT POSSIBILITIES:	29
WHAT COULD HAVE BEEN DONE AS WELL?	29
USING GITHUB FURTHER:	29
MANAGING COMMUNICATION THROUGH SLACK:	29
GETTING EXTERNAL FEEDBACK SOONER:	29
WHAT COULD HAVE BEEN IMPROVED?	30
ASSUMPTIONS WE COULD HAVE WORKED BEYOND	30
EXTRA FEATURES	30

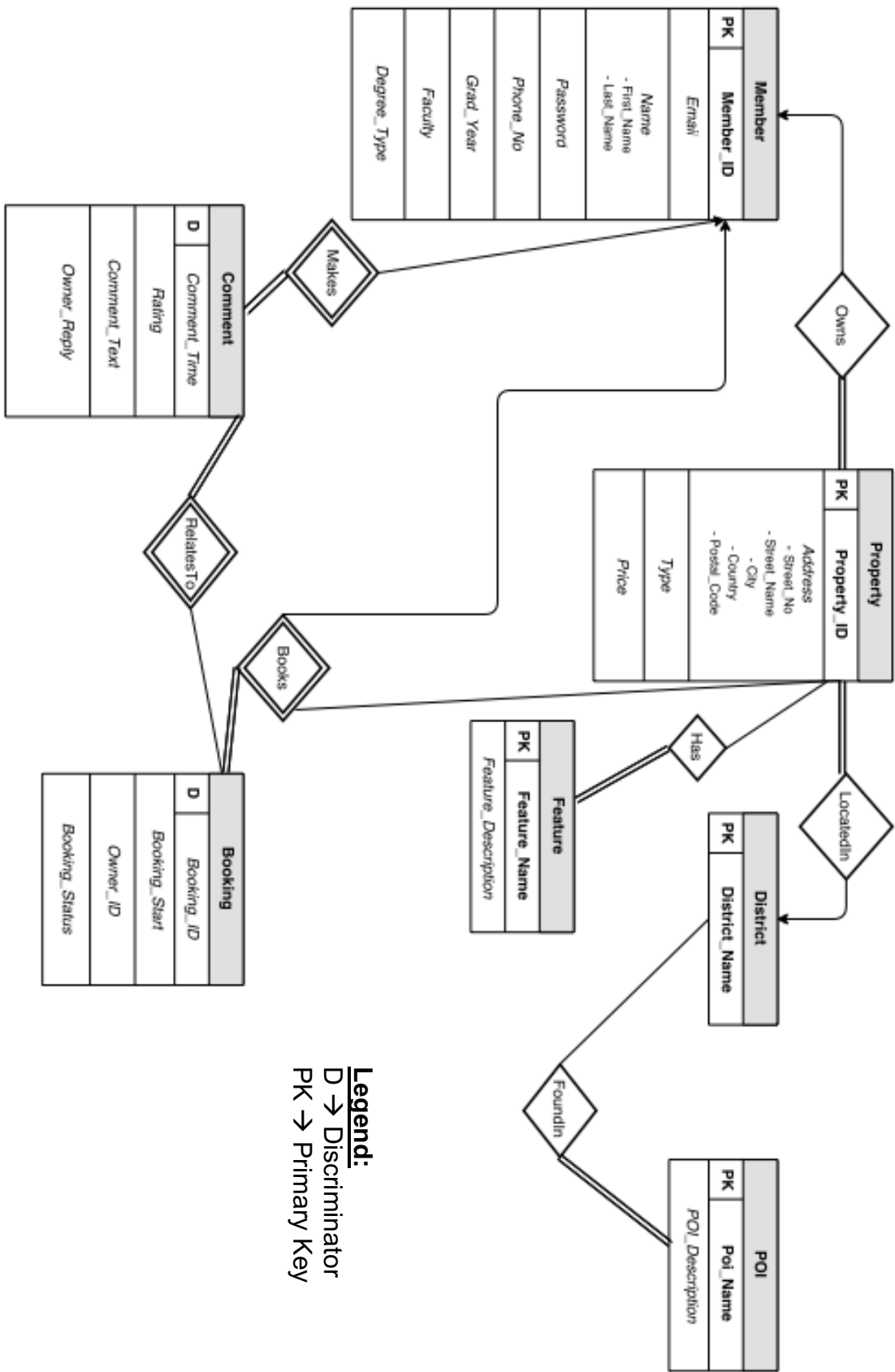
USER GUIDE	32
LOGGING IN & SIGNING UP	32
CONSUMER DASHBOARD	33
SUPPLIER DASHBOARD	34
ADMIN GUIDE	36

Continued Assumptions

- 1) We have used Member_ID as the primary key for our Member table, restricting each email to being associated with one account on QBnB. We are assuming since this is a Queen's alumni database we will not have an integer overflow with the number of registered users.
- 2) Users will not enter invalid information or forms that are incomplete when using our application.
- 3) All Properties in the Property table must be owned by some member of QBnB.
- 4) One member may own multiple properties, but multiple members cannot own the same property.
- 5) Each property belongs to one and only one district, but a district may encompass multiple properties.
- 6) All bookings must be approved (or denied). Users can only cancel bookings if they are "pending". All bookings are for one week in duration.
- 7) The member that owns a rentable property is able to approve (or deny) any bookings, and all bookings need only be approved once.
- 8) Owner_ID corresponds to the Member_ID of the owner of the property in question.
- 9) Any member may comment as many times as they want on a property, but the owner of the property is the only user who may reply directly to those comments on their property.
- 10) In order to comment on a property, a member must have booked it at some point in time. The property being commented on must exist.
- 11) Administrator accounts will be limited to five, and will be the first five accounts created in the Member table (Member_IDs 1, 2, 3, 4 and 5).
- 12) When leaving feedback on a property, a comment is required, but a rating (out of five) is optional.
- 13) An average rating will only be shown if the property being shown has at least one rating given to it.
- 14) When in the user dashboard, administrators will have the same abilities as an equivalent user (consumer if they do not own any properties or supplier if they own at least one).
- 15) Any fields can be updated by a member except for the Member_ID.
- 16) A property is available for booking if and only if there is no booking for it listed at that time, or the booking(s) that is/are currently made on it are listed as rejected or pending for that time.
- 17) Any fields can be updated to a property except for the Property_ID.
- 18) When deleting an accommodation, the comments and bookings associated with it are also deleted.
- 19) When deleting a user account, all properties and the comments and bookings with those properties are deleted.

- 20) Due to the fast-paced nature of our version of QBnB, users are only able to book properties in the span of the next month.
- 21) The only districts available are in Toronto for the current version to limit complexity and to keep the district list short.
- 22) Users cannot edit comments; owners are unable to edit their replies.
- 23) There are a limited number of selectable features, and users are unable to create their own custom features, at this time.
- 24) For demonstration purposes, users are able to leave comments before their booking has been completed.

Entity Relationship Diagram



Legend:
D → Discriminator
PK → Primary Key

Relational Schema

```
CREATE DATABASE QBnB;

USE QBnB;
CREATE TABLE Member(
    Member_ID int NOT NULL AUTO_INCREMENT,
    F_Name varchar(20) NOT NULL,
    L_Name varchar(20) NOT NULL,
    Email varchar(50) NOT NULL,
    Phone_No bigint(15) NOT NULL,
    Grad_Year int(4) NOT NULL,
    Faculty varchar(50) NOT NULL,
    Degree_Type char(10) NOT NULL,
    Password varchar(180) NOT NULL,
    PRIMARY KEY (Member_ID),
    UNIQUE (Email)
);

CREATE TABLE District(
    District_Name varchar(50) NOT NULL,
    PRIMARY KEY (District_Name)
);

CREATE TABLE POI(
    District_Name varchar(50) NOT NULL,
    POI_Name varchar (50) NOT NULL,
    POI_Description varchar(500) DEFAULT NULL,
    PRIMARY KEY (District_Name, POI_Name),
    FOREIGN KEY (District_Name) references District(District_Name) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE Property(
    Property_ID int NOT NULL AUTO_INCREMENT,
    Owner_ID int NOT NULL,
    Street_No int NOT NULL,
    Street_Name varchar(30) NOT NULL,
    City varchar(30) NOT NULL,
    Country varchar(30) NOT NULL,
    Postal_Code varchar(10) NOT NULL,
    District_Name varchar(30) NOT NULL,
    Type varchar(15) NOT NULL,
    Price decimal NOT NULL,
    PRIMARY KEY (Property_ID),
    FOREIGN KEY (District_Name) references District(District_Name) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (Owner_ID) references Member(Member_ID) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE Feature(
    Property_ID int NOT NULL,
    Feature_Name varchar(100) NOT NULL,
    Feature_Description varchar(500) DEFAULT NULL,
    PRIMARY KEY (Property_ID, Feature_Name),
    FOREIGN KEY (Property_ID) references Property(Property_ID) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE Booking(
    Booking_ID int NOT NULL AUTO_INCREMENT,
    Property_ID int NOT NULL,
    Member_ID int NOT NULL,
    Owner_ID int NOT NULL,
    Booking_Start datetime NOT NULL,
    Booking_Status char(15) NOT NULL,
    PRIMARY KEY (Booking_ID, Property_ID, Member_ID),
    FOREIGN KEY (Property_ID) references Property(Property_ID) ON UPDATE CASCADE ON DELETE CASCADE,
```

```

FOREIGN KEY (Member_ID) references Member(Member_ID) ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (Owner_ID) references Property(Owner_ID) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE Comment(
Booking_ID int NOT NULL,
Comment_Time timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
Member_ID int NOT NULL,
Rating int DEFAULT NULL,
Comment_Text varchar(500) NOT NULL,
Owner_Reply varchar(500) DEFAULT NULL,
PRIMARY KEY (Booking_ID, Comment_Time),
FOREIGN KEY (Booking_ID) references Booking(Booking_ID) ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (Member_ID) references Member(Member_ID) ON UPDATE CASCADE ON DELETE CASCADE
);

```


Table Dumps

Property

Property_ID	Owner_ID	Street_No	Street_Name	City	Country	Postal_Code	District_Name	Type	Price
1	1	12	Brock Street	Toronto	Canada	M9A 4X6	Entertainment District	Apartment	35
2	2	12	Bay Street	Toronto	Canada	M4A 9X6	Financial District	Loft	3000
3	1	50	Gerrard Street	Toronto	Canada	M9A 4X1	Danforth	Basement	50
4	3	41	Yonge Street	Toronto	Canada	M9A 4X9	North Toronto	Apartment	75
5	4	1891	Quebec Avenue	Toronto	Canada	M6T 4Q9	High Park	House	15

Point of Interest

District_Name	POI_Name	POI_Description	Gerrard Street East	Gerrard India Bazaar	NULL
Annex	Jewish Community Centre	NULL	Harbourfront	Billy Bishop Airport	NULL
Annex	Koreatown	NULL	Harbourfront	Harbourfront Centre	NULL
Annex	University of Toronto	NULL	Harbourfront	Jack Layton Ferry Terminal	NULL
Beaches	Ashbridges Bay	NULL	Harbourfront	Queens Quay	NULL
Beaches	Kew Gardens	NULL	High Park	High Park	NULL
Beaches	The Boardwalk	NULL	High Park	Sunnyside Docks	NULL
Beaches	The Goof	NULL	North Toronto	Chaplin Estates	NULL
Beaches	Woodbine Beach	NULL	North Toronto	Yonge and Eglinton	NULL
Cabbagetown	Riverdale Farm	NULL	Rosedale	The Integral House	NULL
Cabbagetown	The School of Toronto Dance Th	NULL	Rosedale	The Studio Building	NULL
Chinatown	Chinatown Markets	NULL	Scarborough City Centre	Scarborough Town Centre	NULL
Chinatown	Spadina Ave.	NULL	St. Lawrence	Sony Centre for the Performing	NULL
Danforth	Greektown	NULL	St. Lawrence	St. Lawrence Hall	NULL
Danforth	Taste of the Danforth	NULL	St. Lawrence	St. Lawrence Market	NULL
Distillery District	Gooderham and Worts Distillery	NULL	Yonge and Dundas	Eatons Centre	NULL
Distillery District	Mill Street Brewery	NULL	Yonge and Dundas	Four Seasons Centre for the Pe	NULL
Distillery District	Toronto Christmas Market	NULL	York Mills	Don Valley Golf Course	NULL
Entertainment District	Air Canada Centre	NULL	York Mills	York Mills Shopping Centre	NULL
Entertainment District	CN Tower	NULL			
Entertainment District	Ripleys Aquarium	NULL			
Entertainment District	Rogers Centre	NULL			
Financial District	Fairmont Place	NULL			
Financial District	First Canadian Place	NULL			
Financial District	Royal York Hotel	NULL			
Financial District	Trump Tower	NULL			
Financial District	Union Station	NULL			
Forest Hill	Branksome Hall	NULL			
Forest Hill	Eglinton Theatre	NULL			

Comment

Booking_ID	Comment_Time	Member_ID	Rating	Comment_Text	Owner_Reply
1	2016-03-08 11:07:58	2	5	Great little spot very reasonably priced!	Glad to hear! Thank you for your visit!
4	2016-03-08 11:07:58	5	1	Worst experience I have ever had!	NULL
5	2016-03-08 11:07:58	3	4	Great place with an even better view!	NULL

Feature

Property_ID	Feature_Name	Feature_Description
1	1 bathroom	NULL
2	Full kitchen	NULL
3	Full kitchen	NULL
3	Mahogany floors	NULL
4	2 bathrooms	NULL
4	2 kitchens	NULL
4	Study	NULL
5	1 bathroom	NULL
5	2 bath	NULL
5	Full kitchen	NULL

Booking

Booking_ID	Property_ID	Member_ID	Owner_ID	Booking_Start	Booking_Status
1	5	2	4	2016-03-11 12:00:00	Approved
2	4	5	3	2016-06-11 12:30:00	Pending
3	2	3	2	2016-02-14 12:00:00	Rejected
4	4	5	3	2016-06-11 12:30:00	Approved
5	2	3	2	2016-02-14 12:00:00	Approved

District

District_Name ▲ 1
Annex
Beaches
Cabbagetown
Chinatown
Danforth
Distillery District
Entertainment District
Financial District
Forest Hill
Gerrard Street East
Harbourfront
High Park
North Toronto
Rosedale
Scarborough City Centre
St. Lawrence
Yonge and Dundas
York Mills

Member

Member_ID	F_Name	L_Name	Email	Phone_No	Grad_Year	Faculty	Degree_Type	Password
1	QBnB	Admin	admin@qbnb.ca	6135336666	2000	Computing	BComp	admin
2	Vinyas	Harish	v.harish@queensu.ca	6477675831	2017	Computing	BComp	vin
3	Zac	Baum	zac.baum@queensu.ca	4163203344	2017	Computing	BComp	zac
4	Patrick	Martin	martin@cs.queensu.ca	6131113333	2000	Computing	BComp	cisc332prof
5	Laura	Brooks	laura.brooks@queensu.ca	4164359453	2016	Computing	BComp	cisc332ta

SQL Statements & Sample Outputs

Generic Statements - by User

MEMBER

1.

#Register as member

```
INSERT INTO `Member`
(`F_Name`, `L_Name`, `Email`, `Phone_No`, `Grad_Year`, `Faculty`, `Degree_Type`, `Password`, `Member_ID`)
VALUES('value1', 'value2', 'value3', value4, value5, 'value6', 'value7', 'value8', value9)
```

2.

#Update anything in profile

```
UPDATE `Member`
SET attributel = value1, ...
WHERE Member_ID = theirMemberID
```

3.

#Sign a member in

```
SELECT Member_ID
FROM `Member`
WHERE Email = theirEmailID AND Password = theirPassword
```

4.

#Remove their membership

```
DELETE FROM `Member`
WHERE Member_ID = theirMemberID
```

OWNER

5.

#Create new accommodation

```
INSERT INTO `Property`
(`Street_No`, `Street_Name`, `City`, `Country`, `Postal_Code`, `District_Name`, `Type`, `Price`, `Property_ID`, `Owner_ID`)
VALUES (value1, 'value2', 'value3', 'value4', 'value5', 'value6', 'value7', value8, value9)
```

6.

#Update existing accommodation

```
UPDATE `Property`
SET attributel = value1, ...
WHERE Property_ID = thePropertyID
```

7.

#Remove accommodation

```
DELETE FROM `Property`
WHERE Property_ID = thePropertyID
```

8.

#Approve booking

```
UPDATE `Booking`
SET Booking_Status = 'Approved'
WHERE Booking_ID = theBookingID AND Booking_Start = theBookingStart
```

```

9.
#Reject booking
UPDATE `Booking`
SET Booking_Status = 'Rejected'
WHERE Booking_ID = theBookingID AND BookingStart = theBookingStart

10.
#Reply to comment
UPDATE `Comment`
SET Owner_Reply = ownerCommentText
WHERE Member_ID = commentersMemberID AND Booking_ID = theBookingID AND Comment_Time =
theCommentTime

11.
#List all owned properties + comments / ratings
SELECT Street_No, Street_Name, City, Price, Property.Property_ID, Comment_Text, Rating, Owner_Reply
FROM Property
LEFT JOIN (`Comment` NATURAL JOIN `Booking`)
ON Property.Property_ID = Booking.Property_ID

12.
#Add feature to property
INSERT INTO `Feature`(Property_ID, Feature_Name, Feature_Description)
VALUES (Value1, "Value2", "Value3")

CONSUMER

13.
#Search by district
SELECT DISTINCT Street_No, Street_Name, City, Country, District_Name, Type, Price, Property_ID
FROM Property
WHERE District_Name = selectedDistrict

14.
#Search by type
SELECT DISTINCT Street_No, Street_Name, City, Country, District_Name, Type, Price, Property_ID
FROM Property
WHERE Type = selectedType

15.
#Search by features
SELECT DISTINCT Street_No, Street_Name, City, Country, District_Name, Type, Price, Property_ID
FROM Property NATURAL JOIN Feature
WHERE Feature_Name = selectedFeature

16.
#Search by price
SELECT DISTINCT Street_No, Street_Name, City, Country, District_Name, Type, Price, Property_ID
FROM Property
WHERE Price <= maxPriceEntered AND Price >= minPriceEntered

17.
#List all ratings and comments for a listing
SELECT DISTINCT Property_ID, Comment_Text, Owner_Reply, Rating
FROM Comment NATURAL JOIN Booking
WHERE Booking.Property_ID = thePropertyID
ORDER BY Comment_Time

18.
#List availability of rental
SELECT Booking_ID, Booking_Status
FROM Booking
WHERE Booking_Start = currentWeekStart AND Property_ID = thePropertyID AND (Booking_Status =
"Approved")

```

19.

```
#Show owner details
SELECT DISTINCT F_Name, L_Name, Email, Phone_No
FROM Member NATURAL JOIN Property
WHERE Member_ID = ownerOfSelectedProperty
```

20.

```
#Place booking request
INSERT INTO `Booking` (`Property_ID`,`Booking_Start`,`Booking_Status`,`Member_ID`,`Owner_ID`)
VALUES (value1, value2,'Pending',value3,value4)
```

21.

```
#List all bookings by one member
SELECT Booking_ID, Member_ID, Property_ID, Street_No, Street_Name, City, Booking_Start,
Booking_Status
FROM Booking NATURAL JOIN Property
WHERE Member_ID = theMemberID
GROUP BY Booking_Status
ORDER BY Booking_Start
```

22.

```
#Show details of one booking
SELECT DISTINCT Street_No, Street_Name ,City, District_Name, Type, Price, Booking_Start,
Booking_Status, Owner_ID
FROM Booking NATURAL JOIN Property
WHERE Booking_ID = consumerMemberID AND Property_ID = thePropertyID
```

23.

```
#Add comment and rating for an accommodation
INSERT INTO `Comment` (`Member_ID`,`Property_ID`,`Rating`,`Comment_Text`)
VALUES (commenterMemberID, thePropertyID, ratingValue, 'commentText')
```

24.

```
#Cancel a booking
DELETE FROM `Booking`
WHERE Booking_ID = theBookingID
```

ADMIN

25.

```
#Delete a member and associated listings
DELETE FROM `Member`
WHERE Member_ID = theirMemberID
```

26.

```
#Delete an accommodation
DELETE FROM `Property`
WHERE Property_ID = thePropertyID
```

27.

```
#Summarize bookings and ratings per accommodation
SELECT Booking_Start, Booking_Status, Avg(Rating)
FROM Booking NATURAL JOIN Comment
WHERE Property_ID = thePropertyID
GROUP BY Booking_Status
ORDER BY Booking_Start
```

28.

```
#Summarize bookings and ratings per owner
SELECT Booking_Start, Booking_Status, Avg(Rating)
FROM Booking NATURAL JOIN Comment
WHERE Owner_ID = supplierMemberID
GROUP BY Booking_Status
ORDER BY Booking_Start
```

29.

```
#Summarize booking activity per customer
SELECT Booking_Start, Booking_Status
FROM Booking
WHERE Member_ID = customerMemberID
GROUP BY Booking_Status
ORDER BY Booking_Start
```

Demo Statements - by User

Note that the timestamps on comments may vary from those shown in the original table dumps because the database sometimes had to be dropped as part of the process in order to get the queries working.

MEMBER

1. #Register new member

```
INSERT INTO `Member`
(`F_Name`, `L_Name`, `Email`, `Phone_No`, `Grad_Year`, `Faculty`, `Degree_Type`, `Password`)
VALUES ('Elon', 'Musk', 'e.musk@tesla.com', 4567866787, 1985, 'Commerce', 'BComm', 'money')
```

Member_ID	F_Name	L_Name	Email	Phone_No	Grad_Year	Faculty	Degree_Type	Password
1	QBnB	Admin	admin@qbnb.ca	6135336666	2000	Computing	BComp	admin
2	Vinyas	Harish	v.harish@queensu.ca	6477675831	2017	Computing	BComp	vin
3	Zac	Baum	zac.baum@queensu.ca	4163203344	2017	Computing	BComp	zac
4	Patrick	Martin	martin@cs.queensu.ca	6131113333	2000	Computing	BComp	cisc332prof
5	Laura	Brooks	laura.brooks@queensu.ca	4164359453	2016	Computing	BComp	cisc332ta
6	Elon	Musk	e.musk@tesla.com	4567866787	1985	Commerce	BComm	money

2. #Update anything in profile

```
UPDATE `Member`
SET Password = 'cash'
WHERE Member_ID = 6
```

6	Elon	Musk	e.musk@tesla.com	4567866787	1985	Commerce	BComm	cash
---	------	------	------------------	------------	------	----------	-------	------

3. #Sign in

```
SELECT Member_ID
FROM `Member`
WHERE Email = 'e.musk@tesla.com' AND Password = 'cash'
```

Member_ID
6

4. #Cancel membership

```
#Note: Elon Musk was still in the table prior to this statement
DELETE FROM `Member`
WHERE Member_ID = 6
```

Member_ID	F_Name	L_Name	Email	Phone_No	Grad_Year	Faculty	Degree_Type	Password
1	QBnB	Admin	admin@qbnb.ca	6135336666	2000	Computing	BComp	admin
2	Vinyas	Harish	v.harish@queensu.ca	6477675831	2017	Computing	BComp	vin
3	Zac	Baum	zac.baum@queensu.ca	4163203344	2017	Computing	BComp	zac
4	Patrick	Martin	martin@cs.queensu.ca	6131113333	2000	Computing	BComp	cisc332prof
5	Laura	Brooks	laura.brooks@queensu.ca	4164359453	2016	Computing	BComp	cisc332ta

OWNER

5. #Create new property

```
INSERT INTO `Property`
(`Street_No`, `Street_Name`, `City`, `Country`, `Postal_Code`, `District_Name`, `Type`, `Price`, `Owner_ID`)
VALUES (15, 'Wynd Drive', 'Toronto', 'Canada', 'M7T 6G6H', 'Forest Hill', 'House', 1000, 2)
```

Property_ID	Owner_ID	Street_No	Street_Name	City	Country	Postal_Code	District_Name	Type	Price
1	1	12	Brock Street	Toronto	Canada	M9A 4X6	Entertainment District	Apartment	35
2	2	12	Bay Street	Toronto	Canada	M4A 9X6	Financial District	Loft	3000
3	1	50	Gerrard Street	Toronto	Canada	M9A 4X1	Danforth	Basement	50
4	3	41	Yonge Street	Toronto	Canada	M9A 4X9	North Toronto	Apartment	75
5	4	1891	Quebec Avenue	Toronto	Canada	M6T 4Q9	High Park	House	15
6	2	15	Wynd Drive	Toronto	Canada	M7T 6G6H	Forest Hill	House	1000

6. #Update existing property information

```
UPDATE `Property`
SET Price = 1500
WHERE Property_ID = 6
```

6	2	15	Wynd Drive	Toronto	Canada	M7T 6G6H	Forest Hill	House	1500
---	---	----	------------	---------	--------	----------	-------------	-------	------

7. #Delete property

```
DELETE FROM `Property`
WHERE Property_ID = 6
```

Property_ID	Owner_ID	Street_No	Street_Name	City	Country	Postal_Code	District_Name	Type	Price
1	1	12	Brock Street	Toronto	Canada	M9A 4X6	Entertainment District	Apartment	35
2	2	12	Bay Street	Toronto	Canada	M4A 9X6	Financial District	Loft	3000
3	1	50	Gerrard Street	Toronto	Canada	M9A 4X1	Danforth	Basement	50
4	3	41	Yonge Street	Toronto	Canada	M9A 4X9	North Toronto	Apartment	75
5	4	1891	Quebec Avenue	Toronto	Canada	M6T 4Q9	High Park	House	15

8. #Approve booking

```
UPDATE `Booking`
SET Booking_Status = 'Approved'
WHERE Property_ID = 4 AND Booking_Start = "2016-06-11 12:30:00"
```

2	4	5	3	2016-06-11 12:30:00	Approved
---	---	---	---	---------------------	----------

9. #Reject booking

```
UPDATE `Booking`
SET Booking_Status='Rejected'
WHERE Booking_ID = 2 AND Booking_Start = "2016-06-11 12:30:00"
```

2	4	5	3	2016-06-11 12:30:00	Rejected
---	---	---	---	---------------------	----------


```

10. #Reply to comment
UPDATE `Comment`
SET Owner_Reply = "Glad you liked it!"
WHERE Member_ID = 3 AND Booking_ID = 5 AND Comment_Time = "2016-03-08 22:33:41"

```

5	2016-03-08 22:43:25	3	4 Great place with an even better view!	Glad you liked it!
---	---------------------	---	---	--------------------

```

11. #List all owned properties with comments
SELECT Street_No, Street_Name, City, Price, Property.Property_ID, Comment_Text, Rating,
Owner_Reply
FROM Property
LEFT JOIN (`Comment` NATURAL JOIN `Booking`)
ON Property.Property_ID = Booking.Property_ID

```

Street_No	Street_Name	City	Price	Property_ID	Comment_Text	Rating	Owner_Reply
12	Brock Street	Toronto	35	1		NULL	NULL
12	Bay Street	Toronto	3000	2		NULL	NULL
50	Gerrard Street	Toronto	50	3		NULL	NULL
41	Yonge Street	Toronto	75	4	Worst experience I have ever had!	1	NULL
1891	Quebec Avenue	Toronto	15	5	Great little spot very reasonably priced!	5	Glad to hear! Thank you for your visit!

```

12. #Add feature to property
INSERT INTO `Feature`(Property_ID, Feature_Name, Feature_Description)
VALUES (3, "Basketball court", "The floors of the basketball court are also mahogany!")

```

Property_ID	Feature_Name	Feature_Description
1	1 bathroom	NULL
2	Full kitchen	NULL
3	Basketball court	The floors of the basketball court are also mahoga...
3	Full kitchen	NULL
3	Mahogany floors	NULL
4	2 bathrooms	NULL
4	2 kitchens	NULL
4	Study	NULL
5	1 bathroom	NULL
5	2 bath	NULL
5	Full kitchen	NULL

CONSUMER

```

13. #Search by district
SELECT DISTINCT Street_No, Street_Name, City, Country, District_Name, Type, Price,
Property_ID
FROM Property
WHERE District_Name = "Entertainment District"

```

Street_No	Street_Name	City	Country	District_Name	Type	Price	Property_ID
12	Brock Street	Toronto	Canada	Entertainment District	Apartment	35	1

14.

```
#Search by type
SELECT DISTINCT Street_No, Street_Name, City, Country, District_Name, Type, Price, Property_ID
FROM Property
WHERE Type = "Loft"
```

Street_No	Street_Name	City	Country	District_Name	Type	Price	Property_ID
12	Bay Street	Toronto	Canada	Financial District	Loft	3000	2

15.

```
#Search by feature
SELECT DISTINCT Street_No, Street_Name, City, Country, District_Name, Type, Price,
Property_ID
FROM Property NATURAL JOIN Feature
WHERE Feature_Name = "Basketball court"
```

Property_ID	Owner_ID	Street_No	Street_Name	City	Country	Postal_Code	District_Name	Type	Price
3	1	50	Gerrard Street	Toronto	Canada	M9A 4X1	Danforth	Basement	50

16. #Search by price

```
SELECT DISTINCT Street_No, Street_Name, City, Country, District_Name, Type, Price,
Property_ID
FROM Property
WHERE Price <= 75 AND Price >= 15
```

Street_No	Street_Name	City	Country	District_Name	Type	Price	Property_ID
12	Brock Street	Toronto	Canada	Entertainment District	Apartment	35	1
50	Gerrard Street	Toronto	Canada	Danforth	Basement	50	3
41	Yonge Street	Toronto	Canada	North Toronto	Apartment	75	4
1891	Quebec Avenue	Toronto	Canada	High Park	House	15	5

17. #List all ratings and comments for a listing

```
SELECT DISTINCT Property_ID, Comment_Text, Owner_Reply, Rating
FROM Comment NATURAL JOIN Booking
WHERE Booking.Property_ID = 5
ORDER BY Comment_Time
```

Property_ID	Comment_Text	Owner_Reply	Rating
5	Great little spot very reasonably priced!	Glad to hear! Thank you for your visit!	5

18. #List availability of rental

```
SELECT Booking_ID, Booking_Status
FROM Booking
WHERE Booking_Start = "2016-06-11 12:30:00" AND Property_ID = 4 AND (Booking_Status =
"Approved")
```

4	Approved
---	----------

19. #Provide owner details

```
SELECT DISTINCT F_Name, L_Name, Email, Phone_No
FROM Member NATURAL JOIN Property
WHERE Member_ID = 2
```

F_Name	L_Name	Email	Phone_No
Vinyas	Harish	v.harish@queensu.ca	6477675831

```
20. #Place booking request
INSERT INTO `Booking` (`Property_ID`, `Booking_Start`, `Booking_Status`, `Member_ID`,
`Owner_ID`)
VALUES (2, '2016-08-17 16:30:00', 'Pending', 5, 2)
```

Booking_ID	Property_ID	Member_ID	Owner_ID	Booking_Start	Booking_Status
1	5	2	4	2016-03-11 12:00:00	Approved
2	4	5	3	2016-06-11 12:30:00	Rejected
3	2	3	2	2016-02-14 12:00:00	Rejected
4	4	5	3	2016-06-11 12:30:00	Approved
5	2	3	2	2016-02-14 12:00:00	Approved
6	2	5	2	2016-08-17 16:30:00	Pending

```
21. #List all bookings for one Member
SELECT Booking_ID, Member_ID, Property_ID, Street_No, Street_Name, City, Booking_Start,
Booking_Status
FROM Booking NATURAL JOIN Property
WHERE Member_ID = 3
GROUP BY Booking_Status
ORDER BY Booking_Start
```

Booking_ID	Member_ID	Property_ID	Street_No	Street_Name	City	Booking_Start	Booking_Status
3	3	2	12	Bay Street	Toronto	2016-02-14 12:00:00	Rejected
5	3	2	12	Bay Street	Toronto	2016-02-14 12:00:00	Approved

```
22. #Get details on one booking
SELECT DISTINCT Street_No, Street_Name, City, District_Name, Type, Price, Booking_Start,
Booking_Status, Owner_ID
FROM Booking NATURAL JOIN Property
WHERE Booking_ID = 1 AND Property_ID = 5
```

Street_No	Street_Name	City	District_Name	Type	Price	Booking_Start	Booking_Status	Owner_ID
1891	Quebec Avenue	Toronto	High Park	House	15	2016-03-11 12:00:00	Approved	4

```
23. #Comment on booking
INSERT INTO `Comment` (`Booking_ID`, `Member_ID`, `Rating`, `Comment_Text`)
VALUES (6,5,5, 'Never wanted to leave! Wonderful home.')
```

Booking_ID	Comment_Time	Member_ID	Rating	Comment_Text	Owner_Reply
1	2016-03-08 22:33:41	2	5	Great little spot very reasonably priced!	Glad to hear! Thank you for your visit!
4	2016-03-08 22:33:41	5	1	Worst experience I have ever had!	NULL
5	2016-03-08 22:43:25	3	4	Great place with an even better view!	Glad you liked it!
6	2016-03-08 23:46:20	5	5	Never wanted to leave! Wonderful home.	NULL

```
24. #Delete booking
DELETE FROM `Booking`
WHERE Booking_ID = 5
```

Booking_ID	Property_ID	Member_ID	Owner_ID	Booking_Start	Booking_Status
1	5	2	4	2016-03-11 12:00:00	Approved
2	4	5	3	2016-06-11 12:30:00	Pending
3	2	3	2	2016-02-14 12:00:00	Rejected
4	4	5	3	2016-06-11 12:30:00	Approved

ADMIN

25. #Delete member and associated listings

```
DELETE FROM `Member`
WHERE Member_ID = 4
```

Member_ID	F_Name	L_Name	Email	Phone_No	Grad_Year	Faculty	Degree_Type	Password
1	QBnB	Admin	admin@qbnb.ca	6135336666	2000	Computing	BComp	admin
2	Vinyas	Harish	v.harish@queensu.ca	6477675831	2017	Computing	BComp	vin
3	Zac	Baum	zac.baum@queensu.ca	4163203344	2017	Computing	BComp	zac
5	Laura	Brooks	laura.brooks@queensu.ca	4164359453	2016	Computing	BComp	cisc332ta

Property_ID	Owner_ID	Street_No	Street_Name	City	Country	Postal_Code	District_Name	Type	Price
1	1	12	Brock Street	Toronto	Canada	M9A 4X6	Entertainment District	Apartment	35
2	2	12	Bay Street	Toronto	Canada	M4A 9X6	Financial District	Loft	3000
3	1	50	Gerrard Street	Toronto	Canada	M9A 4X1	Danforth	Basement	50
4	3	41	Yonge Street	Toronto	Canada	M9A 4X9	North Toronto	Apartment	75

Property_ID	Feature_Name	Feature_Description
1	1 bathroom	NULL
2	Full kitchen	NULL
3	Full kitchen	NULL
3	Mahogany floors	NULL
4	2 bathrooms	NULL
4	2 kitchens	NULL
4	Study	NULL

Booking_ID	Property_ID	Member_ID	Owner_ID	Booking_Start	Booking_Status
2	4	5	3	2016-06-11 12:30:00	Pending
3	2	3	2	2016-02-14 12:00:00	Rejected
4	4	5	3	2016-06-11 12:30:00	Approved
5	2	3	2	2016-02-14 12:00:00	Approved

Booking_ID	Comment_Time	Member_ID	Rating	Comment_Text	Owner_Reply
4	2016-03-09 22:46:43	5	1	Worst experience I have ever had!	NULL
5	2016-03-09 22:46:43	3	4	Great place with an even better view!	NULL

26. #Delete property

```
DELETE FROM `Property`
WHERE Member_ID = 5
```

Member_ID	F_Name	L_Name	Email	Phone_No	Grad_Year	Faculty	Degree_Type	Password
1	QBnB	Admin	admin@qbnb.ca	6135336666	2000	Computing	BComp	admin
2	Vinyas	Harish	v.harish@queensu.ca	6477675831	2017	Computing	BComp	vin
3	Zac	Baum	zac.baum@queensu.ca	4163203344	2017	Computing	BComp	zac
4	Patrick	Martin	martin@cs.queensu.ca	6131113333	2000	Computing	BComp	cisc332prof
5	Laura	Brooks	laura.brooks@queensu.ca	4164359453	2016	Computing	BComp	cisc332ta

Property_ID	Owner_ID	Street_No	Street_Name	City	Country	Postal_Code	District_Name	Type	Price
1	1	12	Brock Street	Toronto	Canada	M9A 4X6	Entertainment District	Apartment	35
2	2	12	Bay Street	Toronto	Canada	M4A 9X6	Financial District	Loft	3000
3	1	50	Gerrard Street	Toronto	Canada	M9A 4X1	Danforth	Basement	50
4	3	41	Yonge Street	Toronto	Canada	M9A 4X9	North Toronto	Apartment	75

Property_ID	Feature_Name	Feature_Description
1	1 bathroom	NULL
2	Full kitchen	NULL
3	Full kitchen	NULL
3	Mahogany floors	NULL
4	2 bathrooms	NULL
4	2 kitchens	NULL
4	Study	NULL

Booking_ID	Property_ID	Member_ID	Owner_ID	Booking_Start	Booking_Status
2	4	5	3	2016-06-11 12:30:00	Pending
3	2	3	2	2016-02-14 12:00:00	Rejected
4	4	5	3	2016-06-11 12:30:00	Approved
5	2	3	2	2016-02-14 12:00:00	Approved

Booking_ID	Comment_Time	Member_ID	Rating	Comment_Text	Owner_Reply
4	2016-03-09 22:58:14	5	1	Worst experience I have ever had!	NULL
5	2016-03-09 22:58:14	3	4	Great place with an even better view!	NULL

27. #Summarize bookings and ratings per property

```
SELECT Booking_Start, Booking_Status, Avg(Rating)
FROM Booking NATURAL JOIN Comment
WHERE Property_ID = 4
GROUP BY Booking_Status
ORDER BY Booking_Start
```

Booking_Start	Booking_Status	Avg(Rating)
2016-06-11 12:30:00	Approved	1.0000

```

28. #Summarize bookings and ratings per supplier
SELECT Booking_Start, Booking_Status, Avg(Rating)
FROM Booking NATURAL JOIN Comment
WHERE Owner_ID = 4
GROUP BY Booking_Status
ORDER BY Booking_Start

```

Booking_Start	Booking_Status	Avg(Rating)
2016-03-11 12:00:00	Approved	5.0000

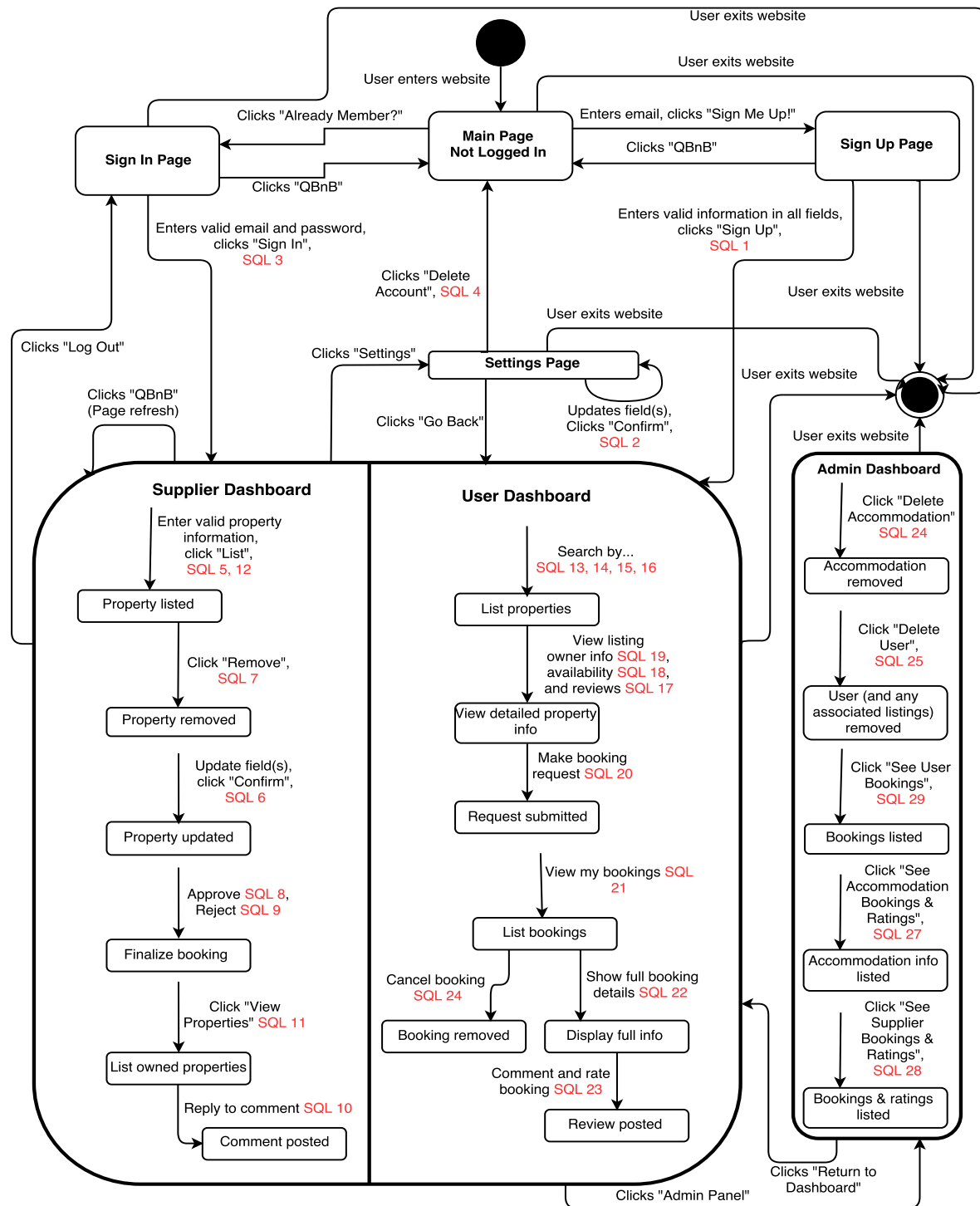
```

29. #Summarize bookings and ratings per customer
SELECT Booking_Start, Booking_Status
FROM Booking
WHERE Member_ID = 2
GROUP BY Booking_Status
ORDER BY Booking_Start

```

Booking_Start	Booking_Status
2016-03-11 12:00:00	Approved

State Machine Diagram



NOTE: The Supplier Dashboard, User Dashboard, Create Property, and Update Property pages are considered to be a part of the entire “Dashboard” state, though they occupy different views in the actual website. Therefore, any state entering or leaving all “Dashboard” states is shared to reduce the number of transitions to states such as the Settings Page state, Exit Website state and Admin Dashboard state.

Discussion

Project Overview

The goal of this project was to design and implement a fully functional, web-hosted database application known as The Queen's Alumnae Bed and Breakfast service (QBnB). QBnB consists of two parts, a server-side (relational database) and client-side (web-application). QBnB will allow for alumnae who intend to travel to a city to be matched with alumnae who are willing to rent out their properties in that city. There are three types of users for QBnB: consumers, owners, and administrators. Owners will register properties on the site with several predefined features in a predefined district within a city. Consumers will use QBnB to search for properties in various districts of the city of interest and then submit a booking request to an owner to rent the property for one week. Owners can then accept or deny the booking and once the booking has been accepted, consumers can leave a comment and rating on the property.

Core Functionality Achieved

We were able to successfully implement all the minimum required functionality for QBnB. The core functionalities for each type of member are listed below:

All Members:

- Register for QBnB
- Update profile
- Cancel membership

Owners:

- List a new property
- Remove or edit an existing property
- List properties
- Submit confirmation or reject a booking
- Reply to a comment

Consumers:

- Search properties by district, type, features, price
- List all the ratings and comments for a property
- List the availability of a property
- Request a booking
- List all a consumer's bookings (past, present, and upcoming)
- Add a comment (and optional rating) for one of their booked properties
- Cancel a booking

Administrators:

- Summarize bookings and ratings per property
- Summarize bookings and ratings per supplier
- Summarize booking activity per customer
- Delete a property and all of its associated bookings and comments
- Delete a member and of their associated properties, bookings, and comments

Design and Implementation Decisions

Design

Landing Page

When new users first enter our website, they find a colourful and welcoming landing page. It was important for us that users found all the information that they needed about QBnB right from the landing page. With our brief description of the QBnB service, “Find Houses. Rent from Alumni.”, right above a registration field and button, users can jump right into our site. If users want more information, scrolling the page provides more details followed by another registration field and button for convenience at the bottom.

If users have already registered in the past, we provide a convenient method of signing-in in the top right-hand corner of the page.

Registering and Signing In

We decided to split the views for the registration and sign-in pages in order to increase the ease of use for the site, and to not clutter a single page with very distinctly different features. However, by providing a link to the sign-in page from the registration page, and vice-versa we ensure that users are not inconvenienced by limited navigation options between pages. Moreover, we wanted to ensure that clicking the QBnB logo allows users to return to our landing page, just in case they wanted to read more about the service before logging-in or registering. Both the registration and sign-in pages are similarly designed and have simple interfaces to ensure that users have a pleasant experience when beginning to interact with our site.

Account Management

To promote easy navigation between pages, we have included a navigation bar at the top of every page. This navigation bar remains anchored at the top of the page so that users will always have access to it. The navigation bar allows users to access a page for updating their account or cancelling their membership, to log out, and it also allows users who are administrators to access the “Admin Dashboard”.

By clicking on the “My Account” button, users access a new page that allows them to view and change all the information that they inputted into their registration form. For simplicity, users do not have the ability to change their Member_ID, as it is kept server-side as a way to organize information across our relations. We decided to display account information in pre-filled forms to allow for convenient editing and the continued use of central design themes through our site. Membership cancellation is handled by a big red button that says exactly what it does; “Delete Account”. We decided to have delete functionality across the service represented in a similar way, with an eye-catching, red colouring to reinforce the gravity of the effects of deleting an account, a booking, or a user’s property.

User Dashboard

Once members log in, they are taken to the user dashboard. Because of the sheer number of functions users can perform, we decided to effectively split the user dashboard into two main views, consisting of three pages. The first is a consumer view and the second is an owner view.

Implementing these views as multiple tables seemed to be the obvious way to represent the amount of data associated with available properties and bookings. HTML and CSS would be used to make the table skeleton, and PHP would be used to pull relevant information from the database using MySQL queries to populate the table. However, it was not realistic to show all of the information associated with properties and bookings in a single row since if we merely displayed something similar to the PHPMyAdmin query output view on a query

with several joins, users may become overwhelmed and the table would also not look aesthetically pleasing. Thus, we show only crucial information in the tables and JavaScript was used to hide and display extra content by clicking on the table row of the property of interest.

In the properties table, clicking on a row opens an extended view of given property where users can see details regarding the owner, features of the property, ratings, comments and points of interest nearby in the district. In this extended view, users can also book a property. By clicking on the “Book me!” button, users are shown a list view where vacant weeks are shown in green and booked weeks are shown in red and are not clickable by the user.

The bookings table was also governed by similar design principles, as we only showed crucial summary information about the booking in the table and do not display other details from users unless they clicked on various table cells to access further information. In order to display this information, we used a popover effect from the Bootstrap framework. Clicking on the address cell gave details on the type of the property and its features while clicking on the owner cell provides the owner’s contact information. Another important feature of the bookings table is the review cell. A message bubble icon indicates that the user has already commented on a booking and can view their comment and rating, if they chose to leave a rating. A pencil and pad icon indicated to the user that they can still leave a comment and rating. The commenting and rating system, like other cells in the table, utilizes a popup that houses a form for typing a comment and slider to leave a rating. With some simple CSS, we were able to animate a 5-star slider to add some colour to the site. If a user has not made any bookings, they see the skeleton of the booking table with a friendly message encouraging them to explore properties and try booking one they are interested in.

Administrator Dashboard

Once administrators log into the website, they can access the “Admin Dashboard” from the navigation toolbar. In fact, we only show the administrator dashboard button to users who are administrators so that normal users would not be confused or mislead in anyway during their interactions with the site. The admin dashboard consists of two main table elements, one summarizing members and the other summarizing properties. Both tables allow the administrator to easily delete a member or property by clicking the red X button.

Summarizing both consumer and owner details were functional requirements of the QBnB service, however we decided to separate these summaries through the use of popovers as to not overwhelm users with two sets of information that are not overtly related.

Clicking on the user’s name summarizes the user’s details as a consumer. The consumer history popover displays the user’s contact information as well as their booking history. We decided to only generate popovers summarizing owner details for those users that owned property, and made it obvious which users were property owners by showing an extra column with a simple “yes” or “no” to indicate their status. Clicking on the “yes” text for members that are owners displays a summary of the member’s role as a property owner with details such as their average rating, number of properties they own, and a history of bookings made on their properties. For members who do own property, but have no bookings or comments, we provide a simple message to inform administrators of that as opposed to showing a blank popover view.

Similarly, we display a summary view of all properties registered on the site. Clicking on the property address summarizes the owner information, its average rating, and a history of bookings made on the property. For properties that have no bookings or comments, we provide a simple message to inform administrators of that as opposed to showing a blank popover

Implementation

Server-Side

Various crucial assumptions, other than those specified in the project outline, were made about how the service worked determined the structure of our relational database. Some of these assumptions include restrictions on:

- Property ownership and property geographical constraints,
- Search, booking, and commenting mechanisms,
- The first five members in the database were administrators,
- The cascading effects of deletions and updates,
- Member and property IDs are unable to be changed.

Based on our assumptions, the relational database was organized so that there were seven main relations, two of which were weak entities, as seen in our Entity Relation Diagram. All relationships were implemented in the relational schema as foreign key constraints. The “on update cascade” and “on delete cascade” constraints were used such that deletions of members, properties, bookings, or comments would be reflected in other relations and thus maintain the consistency of the database. In order to facilitate rapid access to various tables, we decided to implement IDs, such as Member_ID and Property_ID, as integers; as searching on integers is a much faster operation than searching on strings.

Client-Side

We hosted the relational database backend within a web based application front-end guided by the service-oriented (SOA) and model-view-controller (MVC) architectures. Web browsers have become the standard client interface to database servers because they enable large number of users to access the database from anywhere, on multiple platforms. As Javascript and other scripting languages are downloaded and managed by a user's browser, an aesthetically pleasing and highly functional interface can be provided to any and all browser users. A crucial design goal for our iteration of QBnB was to ensure that our web service was aesthetically pleasing, modern, and highly responsive.

Since QBnB offers a range of services for different types of users, following the principles of SOA and MVC allowed us to loosely couple functionality and yet maximize cohesion between various elements. With our relational database essentially serving as our model, we built views and controllers within our web-application to achieve all of the required functionality. Different views, implemented as separate webpages, are provided to users depending on what type of user they were and, by extension, what functionality they were allowed to have.

Technologies and Tools Used

Languages

MySQL

MySQL was the tool we used to build the relational database. It is the most popular client server relational database management system used in the world¹, thus there was lots of documentation and support available online that aided us during the development of QBnB. Moreover, since MySQL is a relational database--it has a strong mathematical basis in relational theory and set theory. Because of this mathematical basis, we were able to reason about why the database worked the way it did and could always distill back down to the

¹ <http://www.oracle.com/us/products/mysql/overview/index.html> -- MySQL, a relational database creation and use tool

mathematical theory when we were faced with a problem. We also enjoyed using MySQL because we developed our ER diagram and relational schema before we programmed the implementation and thus always had a guide to look at when we were stuck on implementing certain features.

PHP

PHP was the scripting language we used in our website to run our MySQL queries and pull information from our server-side relational database. PHP is embedded within HTML, thus the scripts we wrote were ran on the server and returned HTML results which made displaying them in tables. In terms of the relational database driver, we decided to use PHP's MySQLi extension as opposed to PHP data objects (PDO). Aside from the fact that we were only shown how to use MySQLi in class, we knew that we did not need to do any object-relational mapping whatsoever but rather just use simple scripts and thus MySQLi served our purposes well. MySQL within PHP itself is depreciated, however we found it quite easy to transition from the MySQL we learned in class to MySQLi.

HTML and CSS

HTML was used to make the skeleton of the site and add basic site elements such as tables, text, and buttons. Combined with CSS for adding style to the pages, these two languages allowed us to make the client-side "look and feel" of the web application.

JavaScript

JavaScript was used to create the dynamic and animated portions of the web application. Because we used the Bootstrap framework, we needed to include jQuery to facilitate scripting for buttons and create dynamic views. For example, jQuery was needed to display the hidden rows and show extra information in the property table when a user clicked on them.

Technologies & Frameworks Used:

We had to leverage a wide range of tools and frameworks in order to build a minimum viable product for QBnB. With the exception of Microsoft Visio, we really enjoyed using all of the tools and frameworks. Oftentimes, we picked the most popular tools that programmers use to tackle the problems, thus we got to experience first hand why they are so popular.

Microsoft Visio

We originally used Microsoft Visio to draw our ER diagram because generally Microsoft's productivity suite is believed to have best in class functionality. Microsoft Visio is a tool to create professional flow charts and diagrams and has specific support for software engineering design². Although Visio allowed us to make a very aesthetically pleasing ER diagram for the first deliverable, we found its pre-sets confusing to work and it was hard to change any the template diagram to better reflect the structure of ER diagrams that we learned about in class. Moreover, once we made one version of the ER diagram it was extremely difficult to change it. Thus this lead us to move towards another tool, draw.IO for use to prepare diagrams for our second deliverable.

draw.io & Google Drive

After a recommendation from a friend who used draw.io³ in their first deliverable, we decided to move away from Microsoft Visio for our second deliverable. draw.io is an online diagramming tool that is built heavily on

² <https://products.office.com/en-ca/visio/flowchart-software> -- Visio, a mediocre Microsoft Office product

³ <https://www.draw.io> -- draw.io, an interactive diagram designer

Google Drive that allowed us to make an aesthetically pleasing, and easily editable, ER diagram and statechart diagram.

Moreover, since we used Google Drive to manage the creation of our deliverables, draw.io served as a perfect tool that integrated with the rest of our technologies and frameworks used. We especially liked being able to work on documents and diagrams concurrently and still maintain version control to keep track of changes.

GitHub

Since this project involved a significant amount of programming, we knew from the get-go that maintaining a strong repository and version control system would be crucial to our success. Github⁴ is a web-based git repository system that integrates full revision control and source code management along with project management and networking features. Since our team only consisted of two people, we did not need to use multiple branches due to fairly compact amount of functionality we needed in the service. We structured our Git repository around our deliverables, and also made sure to upload the TA’s feedback onto our repository and any other notes to ensure that we really had everything we needed in one place, fully version controlled. One thing about GitHub that we wished we used was the issue and feature tracking features, something that will be explored in detail in the critical reflection section of this report.

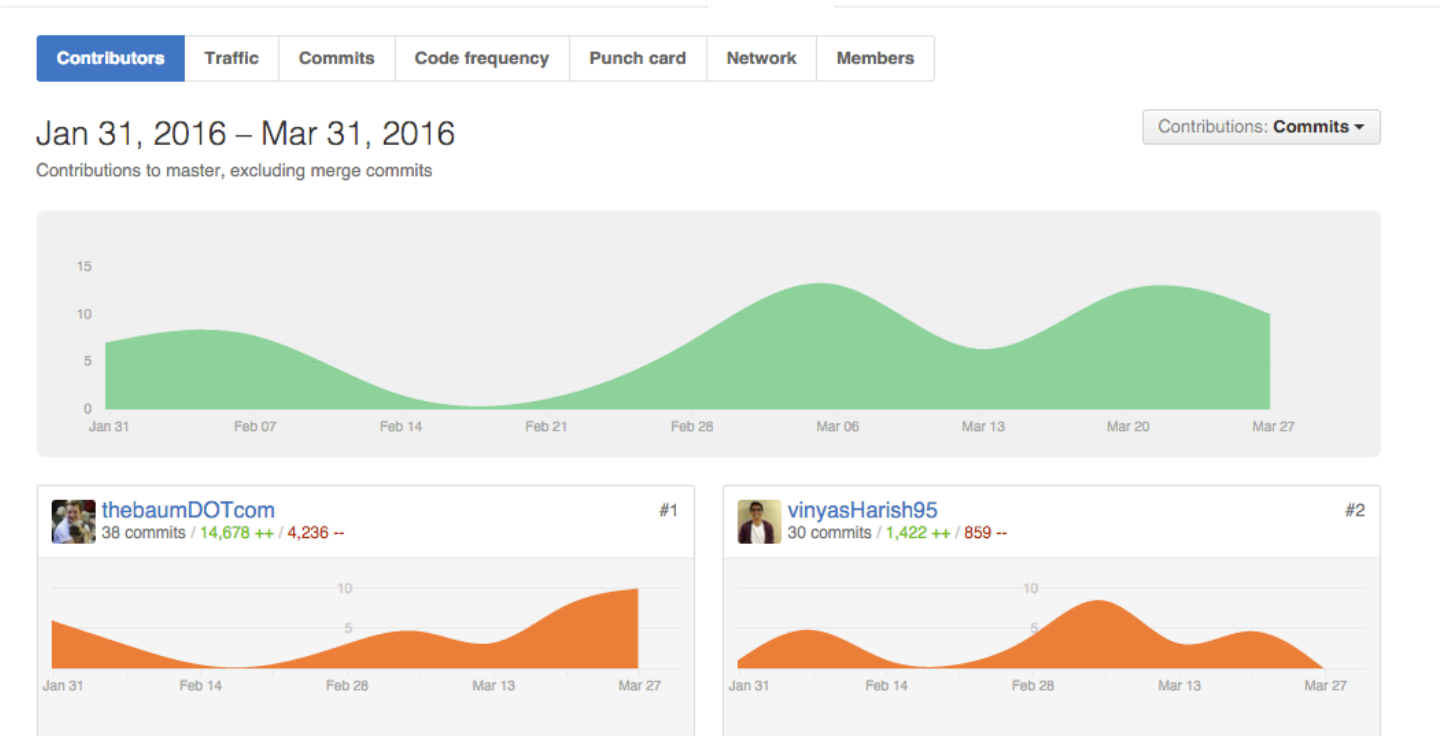


Figure 1: Commit history over the past three months from both members and the overall project. This is here as Zac’s brag sheet. Source: Github. “8,000 of Zac’s lines are the bootstrap.css... Just saying.” Source: Vinyas Harish.

⁴ <https://github.com> -- Github, a git based version control and repository system

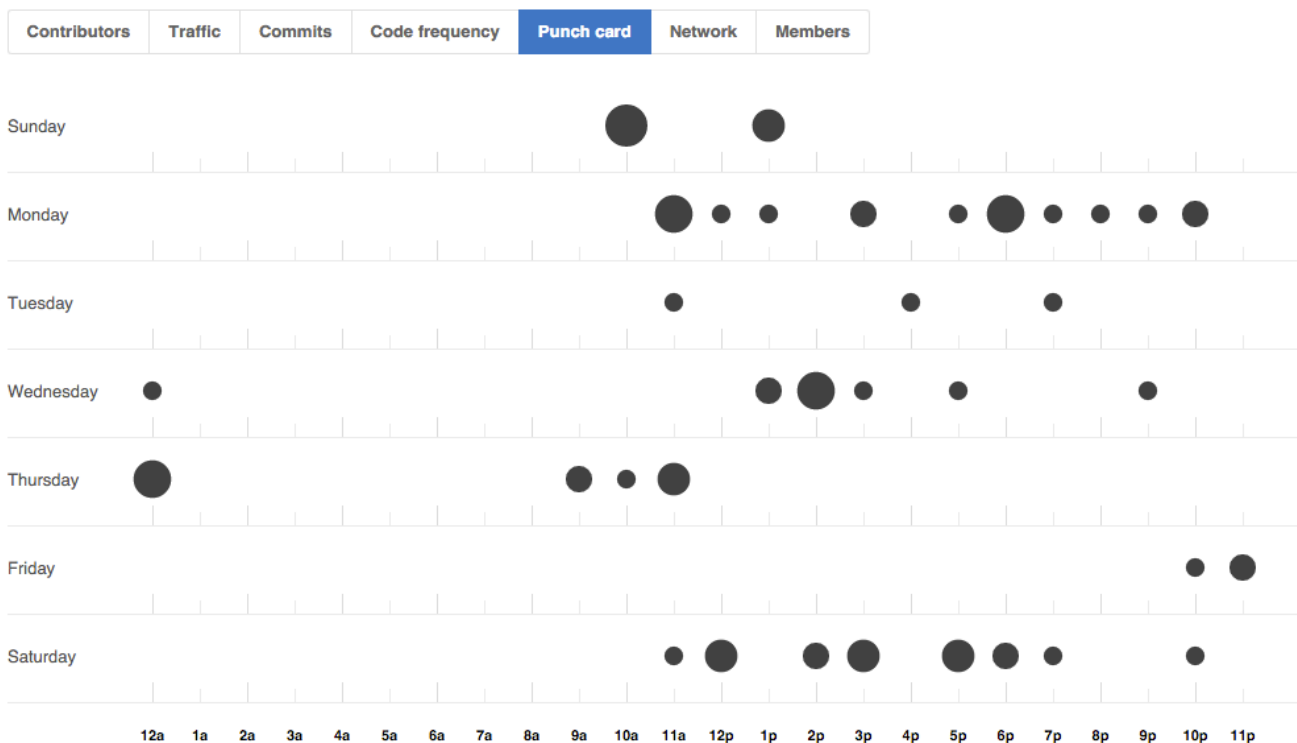


Figure 2: 'Punchcard' of the commit history from both members of the project. Isn't Github fun? No coding on Fridays for us. Source: Github.

Bootstrap Framework

Bootstrap⁵ is an extremely popular, open-source, web design framework that we used to help style and construct our front-end. Some features that made Bootstrap an appealing choice for us during development include its built in support for devices of different shapes and sizes, modern look-and-feel, wide range of custom HTML and CSS components, and jQuery⁶ plugins to create dynamic effects and user interaction possibilities.

Development Environments

When developing QBnB, we used powerful text editors such as Atom⁷ and Sublime Text 2⁸ with multilanguage support, and features such as autocomplete, find and replace, and indentation guides, especially helpful for keeping track of our HTML tags! Since we had to host our back-end in a web application, all functionality testing and debugging could be done in a web browser of our choosing. We decided to test on Google Chrome⁹, since its been the most popular web browser for almost four years with a cross-platform overall usage of 50%¹⁰.

XAMPP

XAMPP¹¹ was the free-to-use PHP development environment that we used to host our server-side relational database for development and testing. Since it hosts the database locally on the user's system, users can use

⁵ <https://getbootstrap.com> -- Bootstrap, an open-source web design framework

⁶ <https://jqueryui.com> -- JQuery, Interactions, effects, widgets and themes built with Javascript

⁷ <https://atom.io> -- Atom, text editor

⁸ <https://www.sublimetext.com/> -- Sublime Text 2, text editor

⁹ <https://www.google.com/chrome> -- Google Chrome, web browser

¹⁰ https://www.wikiwand.com/en/Usage_share_of_web_browsers -- Web browser share among international users

¹¹ <https://www.apachefriends.org/index.html> -- XAMPP, local PHP Development Server

Apache and PHP locally. In order to manage and test our database, we used PHPMyAdmin¹². PHPMyAdmin allows users to perform commonly used operations on databases through the user interface but also have direct control via SQL statements. Moreover, we also found setting up our XAMPP server and connecting to our database to be relatively intuitive and user-friendly. Whenever we ran into difficulties using these tools, we were pleased to find significant amounts of detailed documentation available to assist us.

Problems Faced

Although this project was significant in its size, we did not have many problems in implementing a minimum viable product for QBnB because of our organizational and time management skills, overall work ethic, and good use of software engineering tools and practices.

The sheer amount of functionality needed in the minimum viable product made starting the project intimidating. Moreover, as Vinyas did not have any prior background in website design, we knew that we would have to start the project early in order to successfully design and implement the basic version of QBnB. Because Zac was more well versed in website design from experience in prior classes, we agreed that Vinyas should focus his efforts primarily on the server-side development and written reports while Zac would focus more heavily on building the front-end of the web application. The agreement was that once Zac's early experimentation allowed for the development of various "motifs" of implementing functionality, Vinyas would be able to build upon those motifs to design and implement the administrator dashboard. Before the first deliverable was due, Vinyas spent some time leveraging online education services to learn the basics of HTML and CSS. By doing several Codecademy¹³ introductory courses, Vinyas was able to understand the roles of HTML and CSS to structure and style basic webpages. Doing this extra, self-motivated learning proved to be crucial in ensuring that we were able to deliver our minimum viable product on time. Although Zac was more experienced in developing web applications, he did not have any prior experience in creating dynamic websites using PHP to interact with a server-side relational database. However, as we quickly learned, just about everything that we tried implementing has been done before by someone, somewhere. Thus by leaning on detailed documentation and some tips and tricks from community forums and other sources such as W3Schools¹⁴ and Stack Overflow¹⁵, Zac was able start building an aesthetically pleasing and highly functional web application.

Some agile development techniques were also used in order to ensure our success in the project. We knew from prior courses that thorough planning before a software project enters development has a high return on investment. Thus, from the first week we received the project we met to create a plan of action in how we were going to tackle the largest software development project of our undergraduate education so far. The way the deliverables were structured allowed us to follow an agile development cycle. After a very thorough planning phase consisting of abstracting and creating diagrammatic representations of our service, we gained a solid understanding of what was to be required of our application. Once the planning phase was complete, we entered an intense feature-driven development phase. We focused on tackling one functional requirement at a time and followed development with feature-based testing to find and squash any bugs as we went. During this process we also had very short daily scrums, just to update each other on what we were doing. Finally, we made good use of version control tools to organize and maintain our code.

¹² <https://www.phpmyadmin.net/> -- PHPMyAdmin, database administrative tools

¹³ <https://www.codecademy.com/learn> -- Codecademy, online programming tutorials

¹⁴ <https://www.w3schools.com> -- W3Schools, the worlds largest web developer site

¹⁵ <https://stackoverflow.com/> -- StackOverflow, a collaborative question and answer site for programmers

Critical Reflection and Future Development Possibilities:

Although we were successful in implementing all the needed functionality for having a minimum viable product for QBnB, some critical reflection revealed that there is room for improvement in our development process if we were able to go back and redo the project. Moreover, there is lots of room for future development and improvement.

What could have been done as well?

Using GitHub further:

As mentioned earlier, we created a repository on GitHub in order to manage our project around deliverables and to organize and maintain our code. Since we were only a team of two people, it was not too challenging to keep mental lists of what we needed to do during different stages of development. If we were to go back and redo the project, especially with a larger team, there is room for improvement; and this seems like a very promising place to start. GitHub has a fantastic issues page and tracking system that allows developers to manage todos, bugs, and tickets. We would implement all a target features list of all the functionality that was outlined in the project guidelines and then our development milestones would be available for all team members to review. Especially as we neared completion of the front-end web application, we realized that we had several strange bugs to keep track of. Oftentimes, these bugs were conveyed between us through word of mouth. However, especially if we had a larger team, it would be much more prudent to leverage GitHub's services for bug tracking.

Managing communication through Slack:

Much of our project communication was done through personal channels. This served as a minor inconvenience as we often needed to filter through our personal messages in order to view messages relevant to our project. Using a service such as Slack¹⁶ would have been a much better way to integrate our communications for the project. Channels would have been set-up for each deliverable and each page on our website, allowing us to index and later search through our communications. Slack also features integration with GitHub and Google Docs, allowing us to further integrate our development practices throughout our project lifecycle.

Getting external feedback sooner:

Although we generally had lots of time to complete the project, we found that there were a few weeks wasted while getting feedback between Deliverables 1 and 2. If we consulted a TA, Professor Martin, or friends in course before Deliverable 1 we would have known that our ER diagram and relational schema were correctly designed from the get-go. Thus time would not have been wasted between deliverables and our development cycle speed would have been increased. However, we decided to visit our TA and get help on addressing suggested changes from Deliverable 1 prior to the due date of Deliverable 2. This was a crucial decision that contributed greatly to our success¹⁷ on the second deliverable and allowed us to begin our software implementation phase of the project much sooner.

¹⁶ <https://slack.com> -- Slack, a messaging application for teams

¹⁷ Graham, you tha real MVP.

What could have been improved?

Although we were able to develop a minimum viable product that was able to deliver all the mandatory functionality, if given more time we could have worked beyond some of our assumptions for a more realistic product. We knew how to tackle many of our stretch goals, however for the interest of time and delivering a finished “as asked for” product, we could not do so.

Assumptions we could have worked beyond

Multiple city and district support

As a simplification, QBnB only works in Toronto. Multi-City support could have been implemented as an extra functionality within the “Admin Dashboard” in the form of a button that opened a form for administrators to enter a new city, districts within that city, and corresponding points of interest.

Make other users administrators

As a simplification we assumed that only the first five users in the database would have administrator privileges. An extra functionality within the “Admin Dashboard” would be to grant administrator privileges to other users. This could be implemented with a button on the member table, and an extra column in our member database that represents if a user is an Administrator or a standard user. When the button is clicked, a query could be run to change the value in the member table.

Make bookings per night as opposed to per week

Another simplification we made was to make bookings weekly. However, it would make more sense to make bookings nightly as with any other hotel or bed and breakfast. The bookings table in our database could be altered to have a booking start and booking end date instead of just a booking start date. Moreover, in our web application we could have implemented the list view of weekly dates as a calendar--allowing users to easily choose start and end dates of their booking.

Ensure that comments and ratings cannot be made prior to completion of the stay

Currently for demonstration purposes, we allow comments and ratings to be made immediately after a booking has been approved. This is not the most realistic feature; thus we could implement a check in our PHP scripts to make sure that the timestamp of the attempted comment/rating occurs after the booking has expired.

Extra features

Having support for uploading pictures

There are several areas of our web application that would highly benefit from having support for uploading pictures. When registering as a new member, users could upload a picture to the site, and could similarly do so when registering a new property. We began to look into adding support for adding pictures towards the end of the project. We realized that it would probably mean adding extra columns to the member and property database to store a file path to the picture that they uploaded--however it was too challenging to actually implement this given our time constraints.

Warnings when deleting a property or member

QBnB administrators have a lot of power to delete members and properties, as well as all their associated bookings and comments. However, we currently do not provide a warning to administrators when they click the red ‘x’ button to remind them of the gravity of their action. We briefly experimented with extra JavaScript

scripts to create popups with warning messages, but unfortunately could not get it working in time to integrate in our demo-ready web application.

Showing descriptions on the property page

Although we have a dedicated portion of our relational database for storing descriptions of a property's features, we do not display this information anywhere on our site. We were debating showing these details, however could not determine a good way of doing so without cluttering the view. We faced a similar issue when considering points of interest nearby a property.

Having more detailed user profiles for public viewing

Another stretch goal we were hoping to implement would showing detailed profile views of other users. Clicking on their name would open a popover or modal view displaying not only their contact information, but their degree program and year of completion, an optional description of what sorts of activities they were involved with at Queen's (e.g. AMS, bands, varsity sports, etc.) and a profile picture. However, due to issues with uploading pictures to the database and not wanting to clutter user views we could not determine the best way of implementing this feature on time to include in our product.

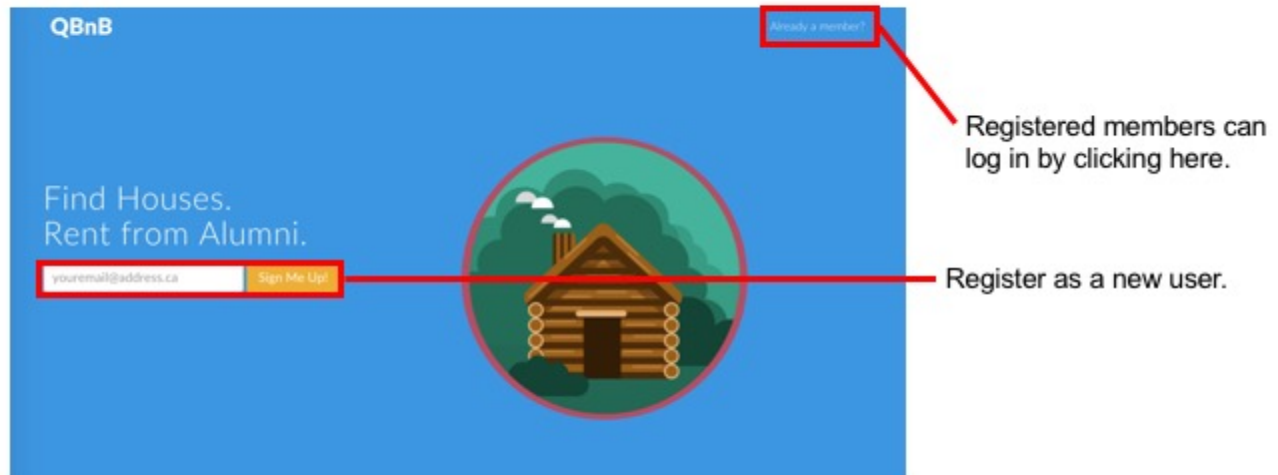
Adding custom features to a property

A final stretch goal we were hoping to implement was the option of users to add custom features to their property. We were thinking of having an extra form on the property management page, but decided against it to avoid cluttering the page and the dropdown menu that is automatically generated from a list of unique features.

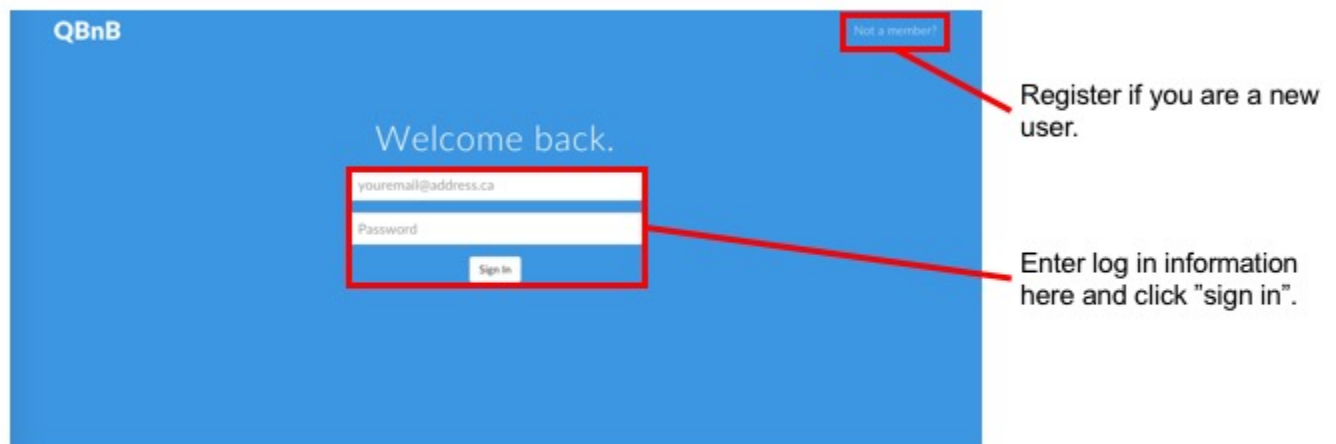
User Guide

Logging in & signing up

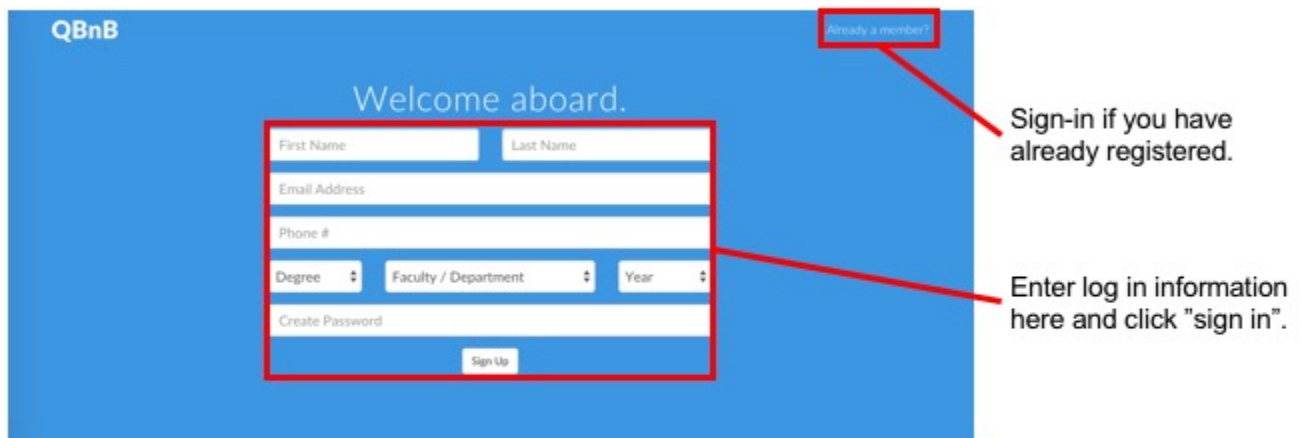
QBnB landing page:



Log in screen for returning members:



Sign up screen for new members:



Consumer Dashboard

Main Consumer page:

The screenshot shows the QBnB main consumer page. At the top, there is a navigation bar with 'Welcome Patrick', 'View & Create Listings', and 'Settings'. Below this is a search bar with the text 'What are you looking for?'. The search bar contains several dropdown menus: 'Any District', 'Any Type', 'Any Feature', 'Price', and 'SCAD'. There are 'Search' and 'Reset' buttons. Below the search bar, there is a table titled 'Available properties:'. The table has columns for Address, District, City, Type, and Price. The first row is highlighted with a red box, and a red arrow points to it with the text 'Click on row to view detailed property information.' Another red arrow points to the 'View & Create Listings' link in the top navigation bar with the text 'Access owner dashboard.' A third red arrow points to the search bar with the text 'Filter list of properties displayed by selecting options from drop-down menu or filling the form and clicking "Search".'

Address	District	City	Type	Price
123 Kite Crescent	Chesham	Toronto	Apartment	\$1100/Week
30 Gerrard Street Unit 2	Danforth	Toronto	Apartment	\$100/Week
12 Brock Street	Entertainment District	Toronto	Townhouse	\$25/Week
12 Bay Street	Financial District	Toronto	Loft	\$300/Week
41 Yonge Street	North Toronto	Toronto	Apartment	\$750/Week
501 Johnson Street	St. Lawrence	Toronto	House	\$400/Week

Update account settings or log-out:

The screenshot shows the user profile menu. It has a blue background and contains the text 'Welcome Patrick', 'View & Create Listings', and 'Settings'. Below this, there is a 'My Account' button and a 'Log Out' button. A red arrow points to the 'Settings' link with the text 'Registered members can log in by clicking here.' Another red arrow points to the 'Log Out' button with the text 'Log-out from the site.'

View extra details on selected property:

The screenshot shows the QBnB property details page. It has a blue header with 'QBnB' and 'Welcome Patrick'. The main content area shows the property details for '12 Bay Street'. It includes the address, district, city, type, and price. There is a 'Book me!' button. Below this, there is a calendar view showing the availability of the property. The calendar shows weeks where the property is unavailable (in red) and weeks where the property is available (in blue). A red arrow points to the 'Book me!' button with the text 'Click "book me" to open list view of dates.' Another red arrow points to the calendar with the text 'Weeks where the property is unavailable are displayed in red.' A third red arrow points to the calendar with the text 'Weeks where the property is available are displayed in blue and are buttons. Click to make a booking request.'

Address	District	City	Type	Price
12 Bay Street	Financial District	Toronto	Loft	\$300/Week

Owner: Vinay Harsh
Email: vharsh@queensu.ca
Phone: 5477675831

Address: 12 Bay Street, M5A 9K6, Toronto
District: Financial District
Nearby: Fairmont Place, First Canadian Place, Royal York Hotel, Drumpf Tower, Union Station
Feature: 1 Bathroom, 1 Bedroom, 1 Kitchen, 2+ Parking Spots, Balcony

No ratings yet!

Your bookings:

Address	Owner	Start Date	Review	Status	Cancel
12 Bay Street	Vinay Harsh	2016-04-02		Pending	

Property search resetting:

The screenshot shows the QBnB search bar. It has a blue background and contains the text 'What are you looking for?'. The search bar contains several dropdown menus: 'Danforth', 'Apartment', '2 Bedrooms', 'Price', and 'SCAD'. There are 'Search' and 'Reset' buttons. A red arrow points to the 'Reset' button with the text 'Reset the search parameters.'

Manage your bookings:

Your bookings:

Address	Owner	Start Date	Review	Status	Cancel
12 Bay Street	Vinay Harish	2016-04-03		Pending	

Click the pencil and pad to write a review.

Click 'x' to cancel the booking.

Make comments on your bookings:

Your bookings:

Address	Owner	Start Date	Review
12 Bay Street	Vinay Harish	2016-04-03	

Make Review

Enter your comment...

★ ★ ★ ★ ★

Post Review

Created by BH & Associates

Enter a comment in this text field.

Click a number of stars to rate the property.

Update user account information:

QBnB

Go Back

Update your account.

Patrick Martin

martin@cs.queensu.ca

6131113333

BComp Computing 2000

Update

Delete Account

Edit fields to change user information, click "Update" to submit changes.

Click "Delete Account" to cancel membership.

Supplier Dashboard

Main Supplier page:

QBnB

Welcome Patrick View & Create Bookings Settings

My listings: [Create New Listing](#)

Address	City	Price	Settings
1891 Quebec Avenue	Toronto	\$200/Week	
8208 Queens Quay	Toronto	\$200/Week	
6 Donlands Avenue	Toronto	\$375/Week	
4001 Queen Street West	Toronto	\$365/Week	
90948 Chestnut Street	Toronto	\$525/Week	

Listed bookings:

Address	User	Booking Start	Status
1891 Quebec Avenue	Zac Baum	2016-03-27	Approved
6 Donlands Avenue	Vinay Harish	2016-04-03	Approved
4001 Queen Street West	Vinay Harish	2016-04-10	Approved
8208 Queens Quay	Vinay Harish	2016-04-10	Approved
1891 Quebec Avenue	Zac Baum	2016-04-17	Approved
4001 Queen Street West	Laura Brooks	2016-04-17	Approved

Create a new listing.

Edit a listing.

Delete a listing.

Approve a booking.

Reject a booking.

Replying to renter comments:





View comments:

Address	Commenter	Comment	Rating	Your Reply
1891 Quebec Avenue	Zac Baum	Too many #FeedTheBarn posters in this neighbourhood... Other than that, awesome!	4/5	
1891 Quebec Avenue	Zac Baum	Great place! Will be coming back soon.	5/5	Thank you very much!
8208 Queens Quay	Vinys Harsh	Awwsome place with a great view! Thanks, Patrick :)	5/5	
4 Dorlands Avenue	Vinys Harsh	Wonderful place! Thanks!	4/5	
4301 Queen Street West	Vinys Harsh	Located right downtown. And you know what that means... it smells bad and had pigeons living on the roof. AWPUS.	1/5	

Click to reply to a consumer's comments.

Making a reply to a comment:

View comments:

Address	Commenter	Comment	Rating	Your Reply
1891 Quebec Avenue	Zac Baum	Too many #FeedTheBarn posters in this neighbourhood... Other than that, awesome!	4/5	
1891 Quebec Avenue	Zac Baum	Great place! Will be coming back soon.	5/5	Thank you very much!
8208 Queens Quay	Vinys Harsh	Awwsome place with a great view! Thanks, Patrick :)	5/5	
4 Dorlands Avenue	Vinys Harsh	Wonderful place! Thanks!	4/5	
4301 Queen Street West	Vinys Harsh	Located right downtown. And you know what that means... it smells bad and had pigeons living on the roof. AWPUS.	1/5	

Write your reply

Enter your reply...

Post Reply

Enter comment text here and click "Post Reply" button.

Creating a new property:

QBnB

Enter your property info:

Go Back

Street # Street Name

Postal Code City Country

Type District Name \$CAD/Week

☒ 1 Bedroom
 ☒ 2 Bedrooms
 ☒ 3+ Bedrooms

☒ 1 Bathroom
 ☒ 2 Bathrooms
 ☒ 3+ Bathrooms

☒ No Kitchen
 ☒ 1 Kitchen
 ☒ 2 Kitchens

☒ No Parking
 ☒ 1 Parking Spot
 ☒ 2 Parking Spots

☒ Balcony
 ☒ Yard
 ☒ Pool
 ☒ Basement

List Property

Enter in property information and click "List Property".

Updating an existing property:

QBnB

Update your property info:

Cancel

8208 Queens Quay

AQA QAO Toronto Canada

Condo Harbourfront 200

☒ 1 Bedroom
 ☒ 2 Bedrooms
 ☒ 3+ Bedrooms

☒ 1 Bathroom
 ☒ 2 Bathrooms
 ☒ 3+ Bathrooms

☒ No Kitchen
 ☒ 1 Kitchen
 ☒ 2 Kitchens

☒ No Parking
 ☒ 1 Parking Spot
 ☒ 2+ Parking Spots

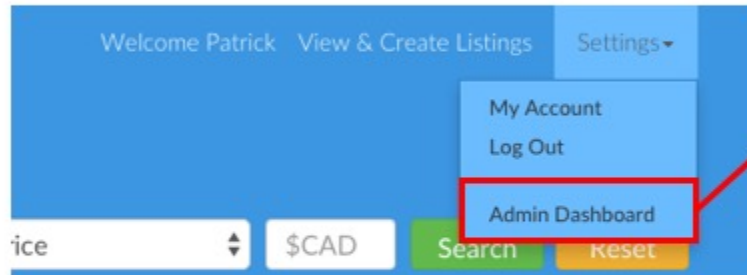
☒ Balcony
 ☒ Yard
 ☒ Pool
 ☒ Basement

Update Property

Enter in edited property information and click "Update Property".

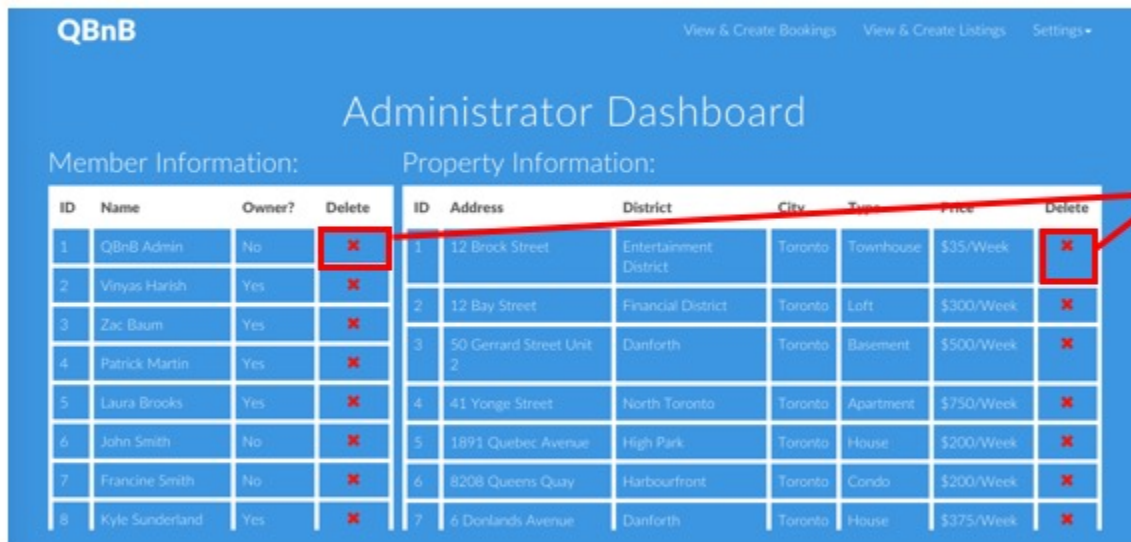
Admin Guide

Accessing the Admin Dashboard:



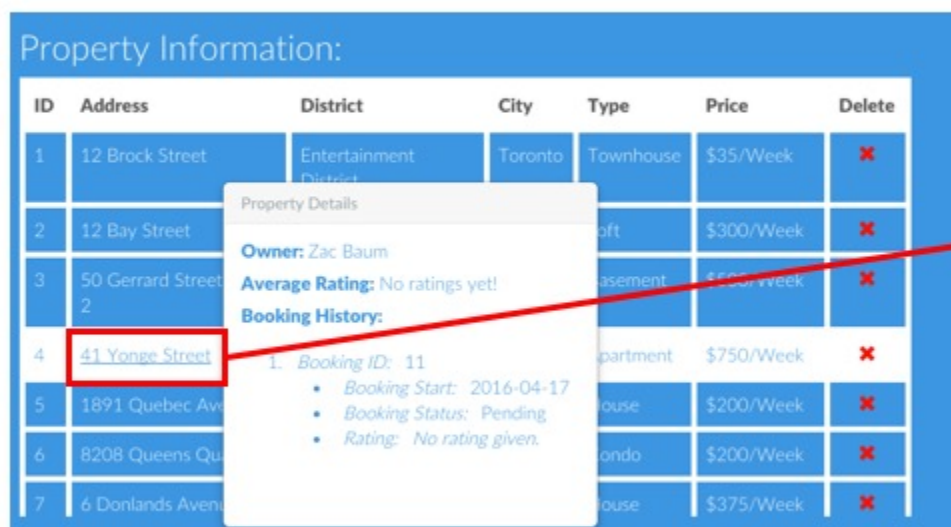
Access the admin dashboard by clicking here. Only displayed for administrators.

Deleting properties and users:



Delete a member or property by clicking on the red 'x'. Note that this deletes all associated properties/bookings/comments.

Viewing property information:



View a detailed report of the property's booking history by clicking on the address.

Viewing member information:

The screenshot shows a table titled "Member Information:" with columns: ID, Name, Owner?, and Delete. A modal titled "Consumer Details" is open over the row for John Smith (ID 6). The modal displays the following information:

- Email:** john.smith@queensu.ca
- Phone:** 6131234567
- Consumer Activity:**
 - Booking ID: 6
 - Booking Start: 2016-04-03
 - Booking Status: Rejected
 - Booking ID: 7
 - Booking Start: 2016-04-24
 - Booking Status: Approved
 - Booking ID: 21

A red arrow points from the text "View a detailed report of the user's activity as a consumer by clicking their name." to the "John Smith" link in the table.

Viewing member owner information, if available.

The screenshot shows the same "Member Information:" table. A modal titled "Owner Details" is open over the row for Zac Baum (ID 3). The modal displays the following information:

- Properties Owned:** 2
- Average Rating:** No ratings yet!
- Booking History:**
 - Property ID: 2
 - Member ID: 2
 - Booking ID: 11
 - Booking Start: 2016-04-17
 - Booking Status: Pending
 - Property ID: 5

A red arrow points from the text "View a detailed report of the user's activity as a property owner by clicking 'Yes'." to the "Yes" link in the "Owner?" column of the table.