

Enterprise Distributed Systems

Peer Reviewed Class Project – airbnb Simulation with auction (Fall 2016)

Project Due Date: 2nd December 2016

Turn in the following on or before November 9, 2016. No late submissions will be accepted!

- An API design document of services provided by the application.

Airbnb History from wiki

Airbnb is an online marketplace that enables people to list, find, then rent vacation homes for a processing fee.[1] It has over 2,000,000 listings in 34,000 cities and 191 countries. Founded in August 2008 and headquartered in San Francisco California

Shortly after moving to San Francisco in October 2007, Brian Chesky and Joe Gebbia created the initial concept for AirBnb & Breakfast during the Industrial Design Conference held by Industrial Designers Society of America. The original site offered short-term living quarters, breakfast, and a unique business networking opportunity for those who were unable to book a hotel in the saturated market.

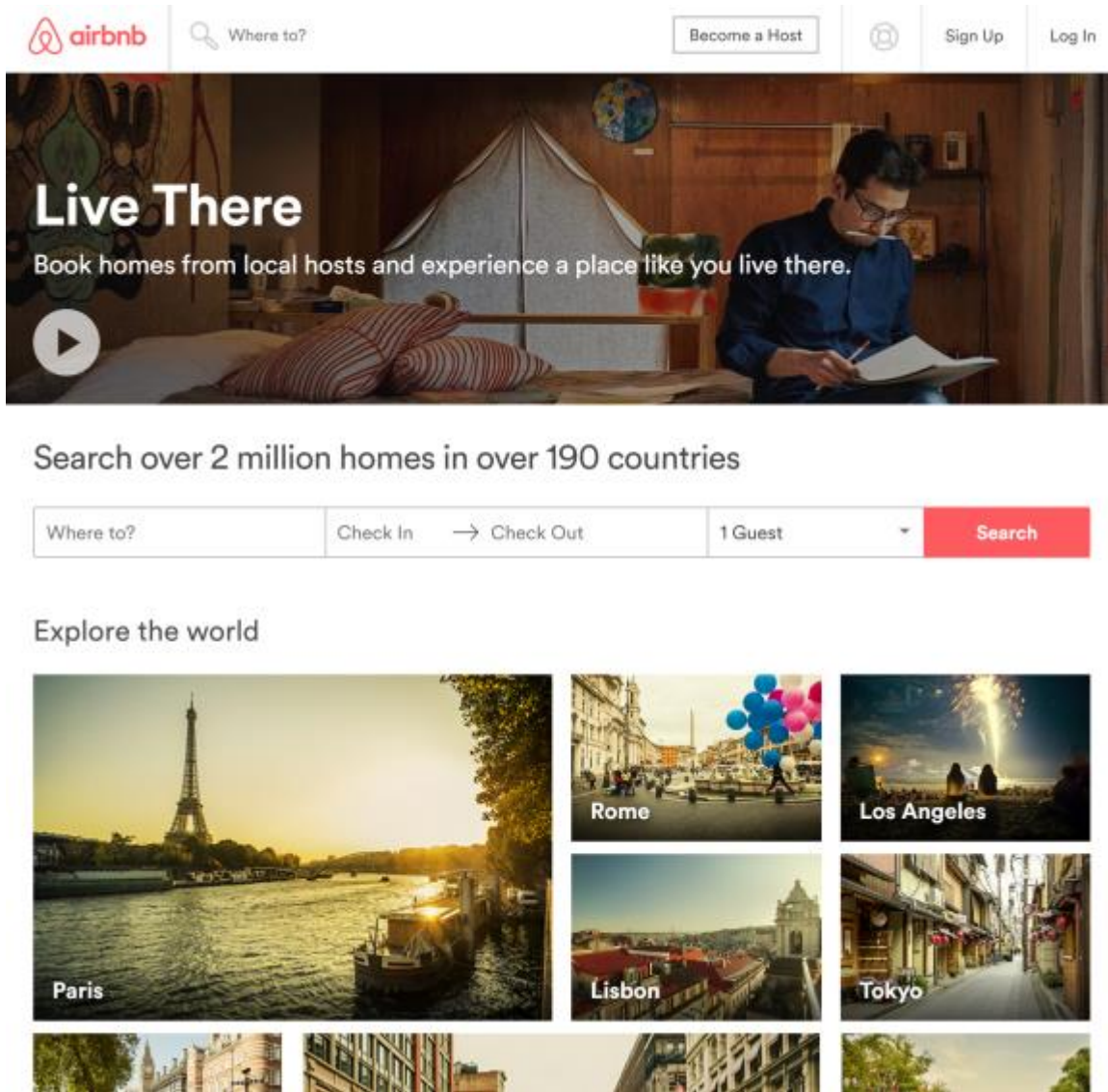
Airbnb is a peer-to-peer accommodation market place that connects hosts (vendors of rooms/accommodations) and travelers via its website. Airbnb enables transactions between these two entities by charging a 'service fee' without directly owning any rooms by itself

Users of the site must register and create a personal online profile before using the site. Every property is associated with a host whose profile includes recommendations by other users, reviews by previous guests, as well as a response rating.

Pricing

Pricing can be determined by the host. Hosts can charge different prices for nightly, weekly, and monthly stays as well as seasonal pricing. We are going to add **auction** feature to Airbnb similar to ebay simulation. A rental unit can be in auction block and subject to bidding among members.

The Titles and Descriptions sections can be used to advertise their space. They can outline house rules or other descriptions regarding the residence. Airbnb allows hosts to publish up to 24 photographs of the place.



Review score

Users of the site may rate host in scale of 1 (bad) to 5 (good); in turn, users may rate host and vice versa. A low rating might diminish the availability and convenience of the service to the user.

Project Overview

In this project, you will design a 3-tier application that will implement the functions of **Airbnb** for house rental services. You will build on the techniques used in your lab assignments to create the system.

In Airbnb prototype, you will manage and implement different types of objects:

- User
- Host
- Administrator
- Analytics
- Bidding

For each type of object, you will also need to implement an associated **database schema** that will be responsible for representing how a given object should be stored in a database.

Your system should be built upon some type of distributed architecture. You have to use message queues as the middleware technology. You will be given a great deal of “artist liberty” in how you choose to implement the system. **These are the basic fields that needed. You can modify this schema according to you need and assign primary keys and foreign keys. Example: Create relation between User and Trips, Host and Properties etc. Also you can divide the tables like Reviews info and User info should be in different table. You can create your own table like Category and make it foreign id in Properties table. Above mentioned are just examples.**

Your system must support the following types of entities:

- **User** – It represents information about an individual host registered on Airbnb. You must manage the following information for this entity:

- ⇒ User ID [SSN Format]
- ⇒ First Name
- ⇒ Last Name
- ⇒ Address
- ⇒ City
- ⇒ State
- ⇒ Zip Code
- ⇒ Phone number
- ⇒ Email
- ⇒ Rating
- ⇒ Reviews
- ⇒ Profile Image
- ⇒ Credit Card details

- **Host** – It represents information about an individual host. You must manage the following state information for this entity:

- ⇒ Host ID [SSN Format]
- ⇒ First Name
- ⇒ Last Name
- ⇒ Address

- ⇒ City
- ⇒ State
- ⇒ Zip Code
- ⇒ Phone number
- ⇒ Email
- ⇒ Credit Card details
- ⇒ Trips History
- ⇒ Rating
- ⇒ Reviews

• **Billing Information** – It represents information about billing for trips. You should provide **Dynamic Pricing** for trip at different times. You must manage the following state information for this entity:

- ⇒ Billing ID [SSN Format]
- ⇒ Date
- ⇒ From Date
- ⇒ To Date
- ⇒ Property Type
- ⇒ Total amount for trip
- ⇒ location
- ⇒ Host ID
- ⇒ User ID

• **Administrator** – It represents information about the administrator of the Uber System. You must manage the following state information for this entity:

- ⇒ Admin ID [SSN Format]
- ⇒ First Name
- ⇒ Last Name
- ⇒ Address
- ⇒ City
- ⇒ State
- ⇒ Zip Code
- ⇒ Phone number
- ⇒ Email

• **Trips** – It represents information about a trip. You must manage the following state information for this entity:

- ⇒ Ride ID [SSN Format]
- ⇒ Property ID
- ⇒ User ID

- ⇒ Host ID
- ⇒ Billing ID

- **Properties** – It represents information about a property. You must manage the following state information for this entity:
 - ⇒ Property ID [SSN Format]
 - ⇒ Host ID
 - ⇒ Category
 - ⇒ Address
 - ⇒ City
 - ⇒ State
 - ⇒ Zip Code
 - ⇒ Quantity
 - ⇒ Description

You may need to create log tracking table to from log files to DB. You need to generate logs into a file when user clicks any place in the web page. There are generally two major logs. Event Logs: record timestamp, userid, click object id, any descriptions. Bidding Logs (periodic logs of bidding process) timestamp, item id, user id, bid amount

Project Requirements

Your project will consist of three tiers:

- The client tier, where the user will interact with your system
- The middle tier/middleware/messaging system, where the majority of the processing takes place
- The third tier, comprising a database to store the state of your entity objects

Tier 1 — Client Requirements

The client will be a node application that allows a user to do the following:

- **User Module/Service:**
 - Create a new User (Reference AirBnb websites for fields informaton)
 - Delete an existing User
 - User can place the bid
 - Change a user's information (name, address, profile image etc) – *This function must support the ability to change ALL attributes*
 - Search for a property based on attributes. You do not have to consider attributes that are not listed above in properties table (Use category table if needed).
 - Display information about a User.
 - Reviews and rating. Must be able to upload photos for properties while reviewing.

- **Host Module/Service:**
 - Create a new Host.
 - Delete an existing Host
 - Dashboard : Able to see analytics activities.
 - Generate Bill for a User (Every Trip).
 - Upload introductory video.
 - Upload images about events during his ride.
 - Approval of requested property by user. (Accept/Denied)


- **Billing Module/Service:**
 - Create a new Bill for each trip.
 - Delete an existing Bill.
 - Search an existing Bill.


- **Admin Module/Service:**
 - Add host to the system.
 - Search for enrolled host area wise. On clicking host, Host profile should get open
 - Review host/customer account
 - Search for a Bill based on attributes (fetch all bills By date, By months)
 - Display information about a Bill.

- **Trips Module/Service:**
 - Create a new trip.
 - Edit an existing trip.
 - Delete an existing trip.
 - History of trips (Display in user's account)
 - History of trips (Display in host's account)

In addition, you may need to create log processing/ log correlation service.

- Display location of properties within the 10 miles of searched location. Pinpoint for price



[Become a Host](#)
[Trips](#)
[Messages](#)
[Help](#)


Dates:

Room type: ☒ Entire home ☐ Private room ☐ Shared room



Price range: \$10 \$1000+


Instant Book: Listings you can book without waiting for host approval. ☐

[Filters](#)

Only 5 listings left for these dates.
 We recommend booking a place soon.

Additional fees apply. Taxes may be added.













- Sample billing report

Customer Receipt

CONFIRMATION CODE: 3WAM4Y

WED, MARCH 07, 2012
RECEIPT # 4024402681

AIRBNB
+1 (415) 800-5959
WWW.AIRBNB.COM/CONTACT

 Chris Coyier NAME	 Austin, TX TRAVEL DESTINATION
 1200 Shelley Ave Austin, TX 78703 United States ACCOMMODATION ADDRESS	 Central Austin Upscale Luxury Home PROPERTY
 Entire home/apt ACCOMMODATION TYPE	 4 GUESTS
 Thu, March 08, 2012 12:00 PM (noon) CHECK IN	 Tue, March 13, 2012 12:00 PM (noon) CHECK OUT
 5 NIGHTS	 \$1000 SECURITY DEPOSIT

Payment Details

ACCOMMODATIONS	\$5000 (\$1000 per night)
AIRBNB SERVICE FEE	\$300
AIRBNB CONCIERGE ^{EXTRA} Premium Guest Services - Details below	Free
TOTAL	\$5300
PAYMENT RECEIVED: THU, OCTOBER 27, 2011 (AMERICAN EXPRESS XXXXXXXXXXXX1008)	\$9388
REFUNDED (AMERICAN EXPRESS XXXXXXXXXXXX1008)	(\$4068)
BALANCE	\$0
DEPOSIT HELD	\$1000



THANKS FOR TRAVELING WITH AIRBNB

99 RHODE ISLAND ST., SAN FRANCISCO, CA, 94103, UNITED STATES

MultiFirefox

- Sample Admin Analysis Report



Admin will have his dashboard. Which will have different types of graphs and charts for the statistics that needs to be displayed.

Graphs Criteria

- 1) Retrieve data from database and show first 10 properties with its revenue/year (Bar, Pie or any kind of graph)
 - 2) City wise revenue/year (Bar, Pie or any kind of graph)
 - 3) 10 hosts who sold maximum number of properties last month with its revenue
- Sample Host Analysis Report



Host will have his dashboard. Which will have different types of graphs and charts for the statistics that needs to be displayed. **Data will be fetched from logging.**

Graphs Criteria

- 1) Graph for clicks per page(Bar, Pie or any kind of graph)
 - 2) Graph for Property click (Bar, Pie or any kind of graph)
 - 3) Capture the area which is less seen.
 - 4) Graph for reviews on properties (Data from database)
 - 5) Trace diagram for tracking one user or a group of users (ex. Users from San Jose, CA)
 - 6) Trace diagram for tracking bidding for an item
- Bidding
 - User should be able to place bid
 - Bid timings will be 4 days
 - After completing the bid user should be able to see pending trip bid in his trip history (No need to implement detailed functionality just Accept/Don't)

You may add any extra functionality you want (optional, not required), but if you do so, you must document and explain the additional features. You still must implement all the required features, however.

The client should have a pleasing look and be simple to understand. Error conditions and feedback to the user should be displayed in a status area on the user interface.

Ideally, you will use an IDE tool to create your GUI.

Tier 2 — Middleware

You will need to provide/design/enhance a middleware that can accomplish the above requirements. You have to implement it using REST based Web Services as Middleware technology. Next, decide on the Interface your service would be exposing. You should ensure that you appropriately handle error conditions/exceptions/failure states and return meaningful information to your caller. Make sure you handle the specific failure cases described below.

ON OR BEFORE the date listed below, you must turn in a document describing the interface your server will use which precisely means that you have to give API request-response descriptions.

Your project will store the state of each host, customers in a relational database. Your project should also include a model class that uses standard modules to access your database. Your entity objects will use the data object to select/insert/update the data in the relational database.

User RabbitMQ as a messaging platform for communication between front-end channels with backend systems.

Tier 3 — Database Schema and Database Creation

You will need to design a database table that will store your relational data. Choose column names and types that are appropriate for the type of data you are using. You will also need a simple program that creates your database and its table. The MySQL Workbench is a very efficient and easy tool for it.

You will need to use MongoDB for storing host introduction videos and images of hosts and customer reviews and updates related to car or host during his ride.

Scalability, Performance and Reliability

The hallmark of any good object container is scalability. A highly scalable system will not experience degraded performance or excessive resource consumption when managing large numbers of objects, nor when processing large numbers of simultaneous requests. You need to make sure that your system can handle many hosts, customer and incoming requests.

Pay careful attention to how you manage “expensive” resources like database connections.

Your system should easily be able to manage 10,000 hosts, 10,000 Users and 100,000 Billing records. Consider these numbers as **minimum** requirements.

Further, for all operations defined above, you need to ensure that if a particular operation fails (for example, a guard allocated two buildings at same time), your system is left in a consistent state. Consider this requirement carefully as you design your project as some operations may involve multiple database operations. You may need to make use of transactions to model these operations and roll back the database in case of failure.

Testing

To test the robustness of your system, you need to design a test harness that will exercise all the functions that a regular client would use. This test harness is typically a command-line program. You can use your test harness to evaluate scalability (described above) by having the test harness create thousands of hosts and customers. Use your test harness to debug problems in the server implementation before writing your GUI.

Other Project Details

Turn in the following on or before the due date. No late submissions will be accepted!

- A title page listing the members of your group
- A page listing how each member of the group contributed to the project (keep this short, one paragraph per group member is sufficient)
- A short (5 pages max) write-up describing:
 - a) Your object management policy
 - b) How you handle “heavyweight” resources
 - c) The policy you used to decide when to write data into the database
- A screen capture of your client application, showing some actual data
- A code listing of your client application
- A code listing of your server implementations for the entity objects
- A code listing of your server implementation for the session object
- A code listing of your main server code
- A code listing of your database access class
- A code listing of your test class
- Any other code listing (utility classes, etc)
- Output from your test class (if applicable)
- A code listing of your database creation class (or script)
- A screen capture showing your database schema
- Observations and lessons learned (1 page max)

You also need to demo the project and to submit a softcopy (via E-Mail to syshim@gmail.com) of all code needed to make your project functional.

The possible project points are broken down as follows:

(One of grading criteria is subjective and relative strength and weakness of your project compared to other projects)

- **40% for basic operation** – Your server implementation will be tested for proper operation of some aspect of your project. Each passed test is worth some point(s) toward the total score, depending on the test. Each failed test receives 0 points.

- **15% for scalability and robustness** – Your project will be judged on its scalability and robustness. I will test your program with thousands of objects; it should not exhibit sluggish performance, and it definitely should not crash. In addition to performance improvement techniques cover in the class, you are required to implement **SQL caching** using Redis and show performance analysis.
- **10% for distributed services** – Divide client requirements into distributed services. Each service will be running on backend connected by RabbitMQ and divide your data into MongoDB and MySQL and provide performance data with justification on results.
- **15% for Analysis report, web/user/item tracking** – Devise your own web pages/user/item tracking using logs and explain why it is effective to analyze a web site and user behavior.
- **10% for the client** – As this project primarily stresses server-side development, the client GUI is not as important. However, I do want to see some sort of GUI developed. You must code a client GUI; I will not accept client-less submissions.
- **10%** for your test class and project write-up

Hints:

- Maintain a pool of DB connections – Remember that opening a database connection is a resource-intensive task. It would be advisable to pre-connect several database connections and make use of them throughout your database access class so you don't continually connect/disconnect when accessing your database.
- Cache entity lookups – To prevent a costly trip to the database, you can cache the state of entity objects in a local in-memory cache. If a request is made to retrieve the state of an object whose state is in the cache, you can reconstitute the object without checking the database. Be careful that you invalidate the contents of the cache when an object's state is changed. In particular, you are required to implement **SQL caching using Redis**.
- Don't write data to the database that didn't change since it was last read.
- Focus **FIRST** on implementing a complete project – remember that a complete implementation that passes all of the tests is worth as much as the performance and scalability part of the project.
- Do not over-optimize. Project groups in the past that have tried to implement complex Optimizations usually failed to complete their implementations.

Exceptions/Failure Modes

Your project **MUST** properly handle the following failure conditions

- Creating Duplicate User/Host
- Addresses (for Person) that are not formed properly (see below)

For more failure conditions see Other Notes below

Other Notes

State abbreviation parameters

State abbreviations should be limited to valid US state abbreviations or full state names. A Google search will yield these valid values. Methods accepting state abbreviations as parameters are required to raise the 'malformed_state' exception (or equivalent) if the parameter does not match one of the accepted parameters. You do not need to handle US territories such as the Virgin Islands, Puerto Rico, Guam, etc.

Zip code parameters

Zip codes should adhere to the following pattern:

[0-9][0-9][0-9][0-9][0-9]

or

[0-9][0-9][0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]

Examples of valid Zip codes:

95123

95192

10293

90086-1929

Examples of invalid Zip codes:

1247

1829A

37849-392

2374-2384

Host/User IDs are required to match the pattern for US social security numbers:

[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9][0-9][0-9]

You may assume that any SSN matching the above pattern is valid (there are some reserved social security numbers starting with 0 that are not valid in real life, but you may consider them to be valid for the purposes of the project). Any method accepting a Host ID is required to raise the 'invalid_host_id' exception (or equivalent) if the supplied parameter does not match the pattern above.

Peer Review

Each team member should write review about other team members except himself/herself. It is confidential which means that you should not share your comments with other members. Peer Review is submitted separately on Canvas.

Extra Credit (10pts)

There will be extra points allocated for creating mobile application apart from web application. You can use any hybrid mobile app framework to generate mobile app from web application code.

Eg. <http://ionicframework.com/>

There is no partial credit for Extra credit. You will get either full marks or none based on your mobile application quality.