Enterprise Distributed Systems

Class Project – AmazonFresh Farmers Market Simulation (Spring 2016)

Project Due Date: 1st May 2016

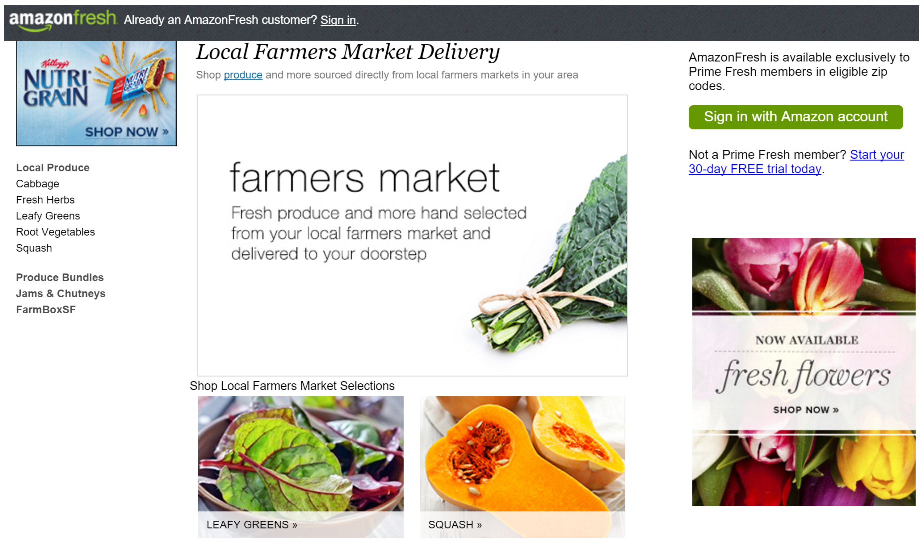
Turn in the following on or before April 8, 2016. No late submissions will be accepted!

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

Project History

Amazon.com, Inc., is the largest Internet-based retailer in the United States.[12]Amazon.com started as an online bookstore.
AmazonFresh Farmers Market offers grocery items for sale, as well as a subset of items from the main Amazon.com storefront. Items ordered are available for home delivery on the same day or the next day, depending on the time of the order and the availability of trucks. Examples of grocery items for sale are wine, pumpkin pie, and precooked turkeys. Check out the farmers market at
https://fresh.amazon.com/Category?cat=localfarmersmarket&ref_=Fr_Gr_AMG_Farm
However it does not allow independent farmers to sell their products.

This project builds **Amazon Farmers Market plus** where local farmers are allowed to sell their fresh produce such as lettuce, tomato, potato, oranges, carrots, etc directly to customers. Here farmers put their descriptions of products and introduction to the farms that includes history, farming methods, video tour, and an owner profile.
Customers can browse many local farmers and items, and make purchases to be delivered next morning. They can check the order progress in the web and also through google maps. Your team needs to build similar site as Farmers Market.


amazonfresh Already an AmazonFresh customer? Sign in.

Local Farmers Market Delivery

Shop produce and more sourced directly from local farmers markets in your area

AmazonFresh is available exclusively to Prime Fresh members in eligible zip codes.

Sign in with Amazon account

Not a Prime Fresh member? Start your 30-day FREE trial today.

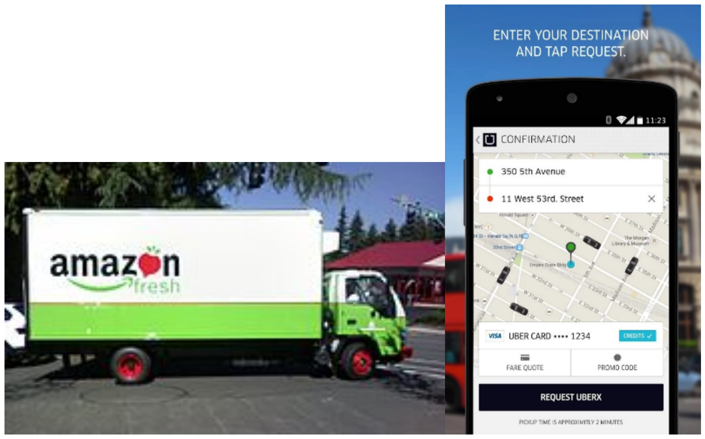
Local Produce
Cabbage
Fresh Herbs
Leafy Greens
Root Vegetables
Squash

Produce Bundles
Jams & Chutneys
FarmBoxSF

farmers market
Fresh produce and more hand selected from your local farmers market and delivered to your doorstep

Shop Local Farmers Market Selections

Once order is submitted, then you can check the progress of the delivery truck through google maps.



Review score

Users of the app may rate quality of products and delivery in scale of 1 (bad) to 5 (good); Hence you maintain two category of rating. A low rating might diminish the availability of produce from the particular farmers.

Project Overview

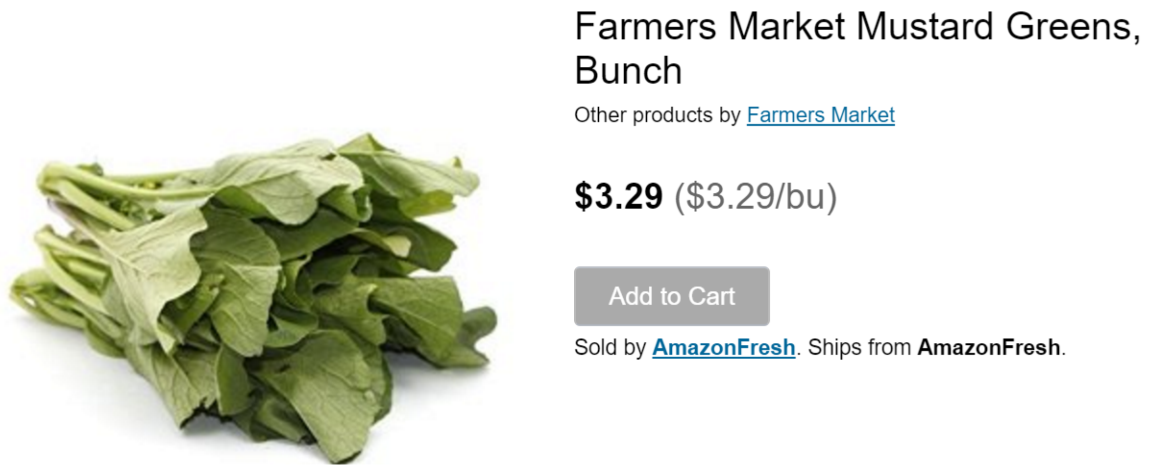
In this project, you will design a 3-tier application that will implement the functions of **AmazonFresh System** for cab services. You will build on the techniques used in your lab assignments to create the system.

In AmazonFresh System, you will manage and implement different types of objects:

- Farmers
- Products

- Trucks
- Customers
- Billing
- Admin
- Trips

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 relational/nosql database.



Farmers Market Mustard Greens, Bunch

Other products by Farmers Market

$3.29 ($3.29/bu)

Sold by **AmazonFresh**. Ships from **AmazonFresh**.

Your system should be built upon some type of distributed architecture. You have to use web services as the middleware technology. You will be given a great deal of "artist liberty" in how you choose to implement the system.

Your system must support the following types of entities:

● **Farmers** – It represents information about an individual farmers registered on AmazonFresh. You must manage the following information for this entity:

⇒ Farmer ID [SSN Format]
⇒ First Name
⇒ Last Name
⇒ Address
⇒ City
⇒ State
⇒ Zip Code
⇒ Phone number
⇒ Email
⇒ Rating
⇒ Reviews
⇒ Own introduction in form of images and video.
⇒ Delivery history

● **Trucks** – It represents information about a delivery truck registered on AmazonFresh. You must manage the following information for this entity:

⇒ Truck ID
⇒ Driver ID [SSN Format]
⇒ First Name
⇒ Last Name
⇒ Address
⇒ City
⇒ State
⇒ Zip Code
⇒ Phone number
⇒ Email
⇒ Car details
⇒ Rides history (location, time)

● **Products** – It represents information about a product on AmazonFresh. You must manage the following information for this entity:

⇒ Product ID
⇒ Farmer ID [SSN Format]
⇒ Name
⇒ Price
⇒ Description
⇒ Reviews
⇒ Ratings

• **Customers** – It represents additional information about an individual customer. You must manage the following state information for this entity:

⇒ Customer ID [SSN Format]
⇒ First Name
⇒ Last Name
⇒ Address
⇒ City
⇒ State
⇒ Zip Code
⇒ Phone number
⇒ Email
⇒ Credit Card details
~~⇒ Purchase History~~

• **Billing Information** – It represents information about billing for produce. You must manage the following state information for this entity:

⇒ Billing ID [SSN Format]
⇒ Date
⇒ Expected Delivery time

⇒ Product ID
⇒ Distance covered
⇒ Total amount for ride
⇒ Source location
⇒ Destination location
⇒ Driver ID
⇒ Customer ID

- **Administrator** – It represents information about the administrator of the AmazonFresh 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 **delivery** trip. You must manage the following state information for this entity:
    ⇒ Trip ID [SSN Format]
    ⇒ Pickup location (Latitude, Longitude)
    ⇒ Drop off location (Latitude, Longitude)
    ⇒ Date/Time
    ⇒ Customer ID
    ⇒ Driver ID
    ⇒ Truck ID

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:

- **Farmers Module/Service:**
  · Create a new Farmer
  · Delete an existing Farmer
  · List all Farmers known by the system
  · Change a farmer's information (name, address, etc) – *This function must support the ability to change ALL attributes*
  · Search for a farmer based on attributes. You do not have to consider attributes that are not listed above.
  · Display information about a farmer.
  · Introduction video of the farmer.

- **Product Module/Service:**
  · Create a new Product
  · Delete an existing Product
  · List all Products known by the system
  · Change a Product's information – *This function must support the ability to change ALL attributes*
  · Search for a product based on attributes. You do not have to consider attributes that are not listed above.
  · Display information about a product.

- **Customer Module/Service:**
  · Create a new Customer.
  · ~~Delete an existing Customer~~
  · List all Customers known by the system
  · Generate Bill for a Customer (Every Ride).
  ·Select delivery time and date
  Post reviews and ratings (reviews may contain images and videos)

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

- **Admin Module/Service:**
  · Add farmer to the system.
  Add product to the system
  · Add customer to the system.
  · Review product/customer account.
  · Show statistics (revenue/day) and total delivery (area wise).
  · Show graphs/charts for different rides per area, per driver, per customer.
  · Search for a Bill based on attributes
  · Display information about a Bill.

- **Trips Module/Service:**
  · Create a new trip.

· Edit an existing trip.
· Delete an existing trip.
· List all trips of a truck.
· List all trips of a driver.
· Show trips statistics as per location.

- Display location of delivery truck from customer location.

Russian Hill
Washington Square
Hyde St
Cable Car Museum
Nob Hill

- Sample Admin Analysis Report for all deliveries from farms to customers

South Bay
Lincoln Park
Sea Cliff
Vista del Mar
Golden Gate Heights
Saint Francis Wood
Balboa Park
Lake Merced Park
Parkmerced
Olympic Golf Club
John McLaren Park
Candlestick Point State Recreation Area
Oakland Naval Supply Center
UBER
The closest driver is approximately 7 mins away
50 Broadway
Set pickup location
Map data ©2012 Goo

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 farmer, 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 product and farmer introduction videos and images of products and farmers and customer reviews and updates on delivery.

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 farmers, 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 products, 10,000 Customers 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 farmers 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:
• **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.
• **25% 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 Radis and show performance analysis.
• **12% 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.
• **5% for dynamic pricing algorithm** – Devise your own dynamic pricing algorithm and explain why it fits to supply and demand AmazonFresh model.
• **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.
• **8%** 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 Radis.

· 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 Driver/Customer
- 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

Farmer/Customer 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 Farmer ID is required to raise the 'invalid_farmer_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 (>8) or none based on your mobile application quality. This means that in order to get scores, most of the functionality including maps must work.