<RESTAURANT WAIT TIME>

DESIGN SPECIFICATION

Version *<1.0>*

<04/10/2016>

1 GENERAL OVERVIEW AND DESIGN GUIDELINES/APPROACH

A. The Azure server side will be hosting an API service which will keep and maintain all the data flowing between server and client.
B. The Client side application will use the RESTAPI call to retrieve the data.

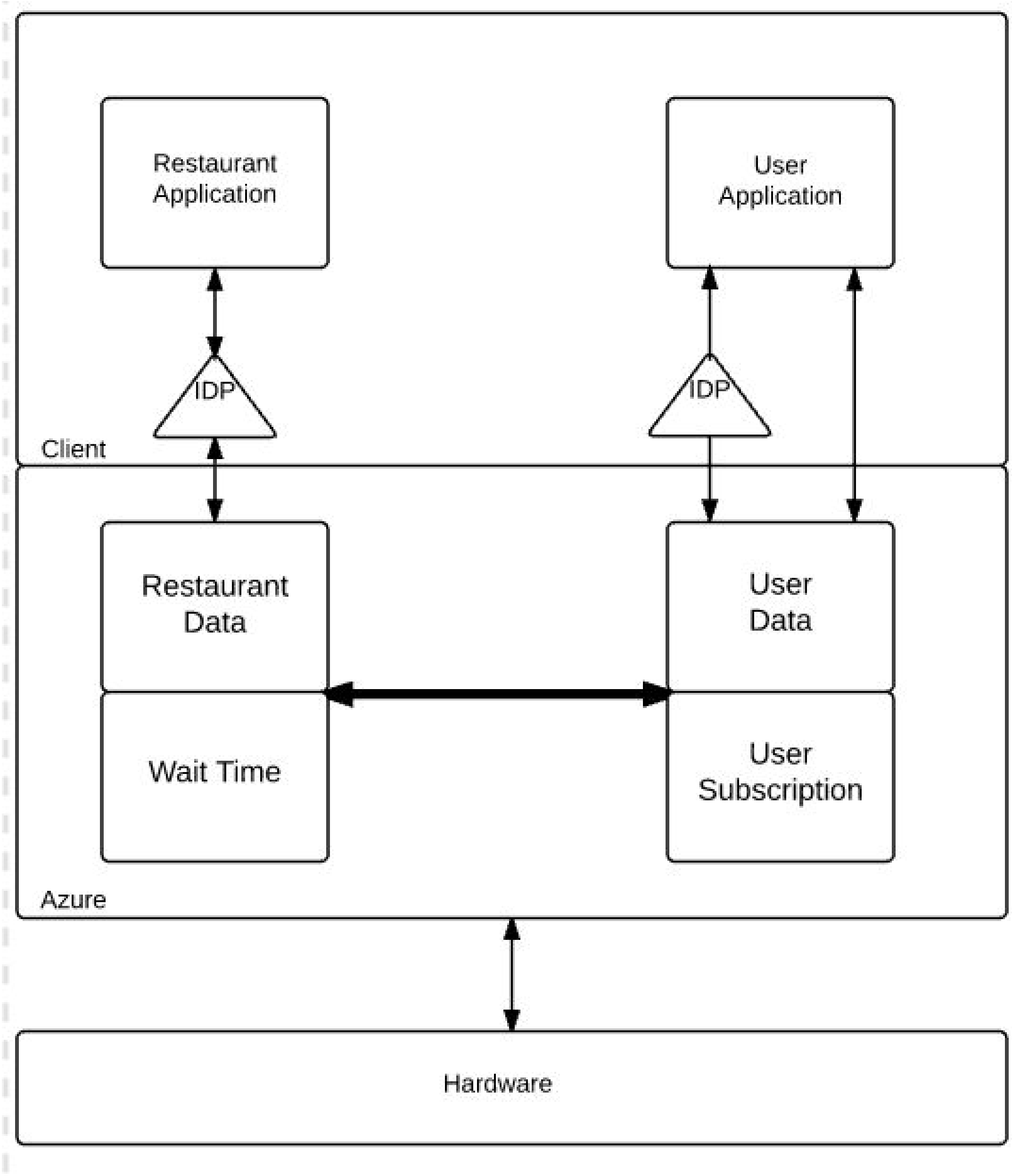
1.1 ASSUMPTIONS / CONSTRAINTS / STANDARDS

- All operation shall be done asynchronously.
- Map Azure WEB API Service REST errors to standard HTTP Error codes.
- All exception must be handled.
- Secure SQL injection attack.
- Secure unauthenticated access to costly resources.
- Does not factor in correlation between user location and traffic condition.
- Client is running on Universal Windows App
- Azure API service always up and ready

2 ARCHITECTURE DESIGN

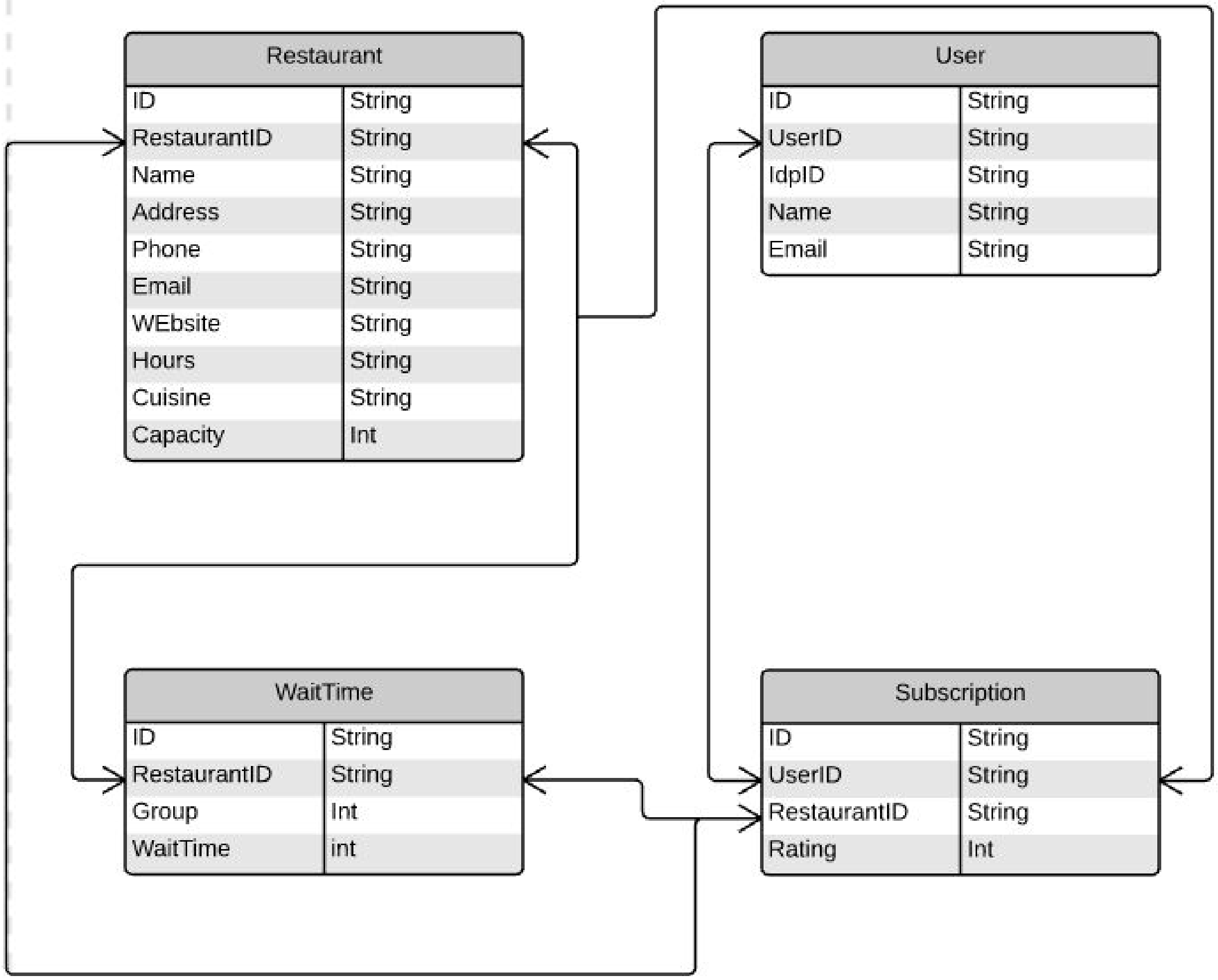
Overall system architecture is client application interact with Azure API service via RESTAPI. Textual data are passing around by json or xml. Image and other was passing as binary blob. Client application will not interact with other client application and Azure API service should be a standalone service.

2.1 High Level Architectural Block Diagram



- On the client side there are 2 application. One for restaurant and one for user
- Restaurant application only accept authenticated user and let the restaurant update restaurant information and waittime. It also allow restaurant to upload menu and image(web job to process and upload to storage) as well as running report (worker roles)
- User application accept anonymous as well as authenticated login. Anonymous user can only view current wait time. Authenticated user can subscribe to a restaurant and get notification on waittime. They can also leave feedback and comments
- Azure is running API APP and process restapi call coming from the client it will also push out notification when waittime changed (push notification and notification hub)
- Azure Redis Cache is being used to cache the user subscription when try to re-pull the wait time.

2.2 Data Structure Class Diagram



Restaurant	
ID	String
RestaurantID	String
Name	String
Address	String
Phone	String
Email	String
WEbsite	String
Hours	String
Cuisine	String
Capacity	Int

User	
ID	String
UserID	String
IdpID	String
Name	String
Email	String

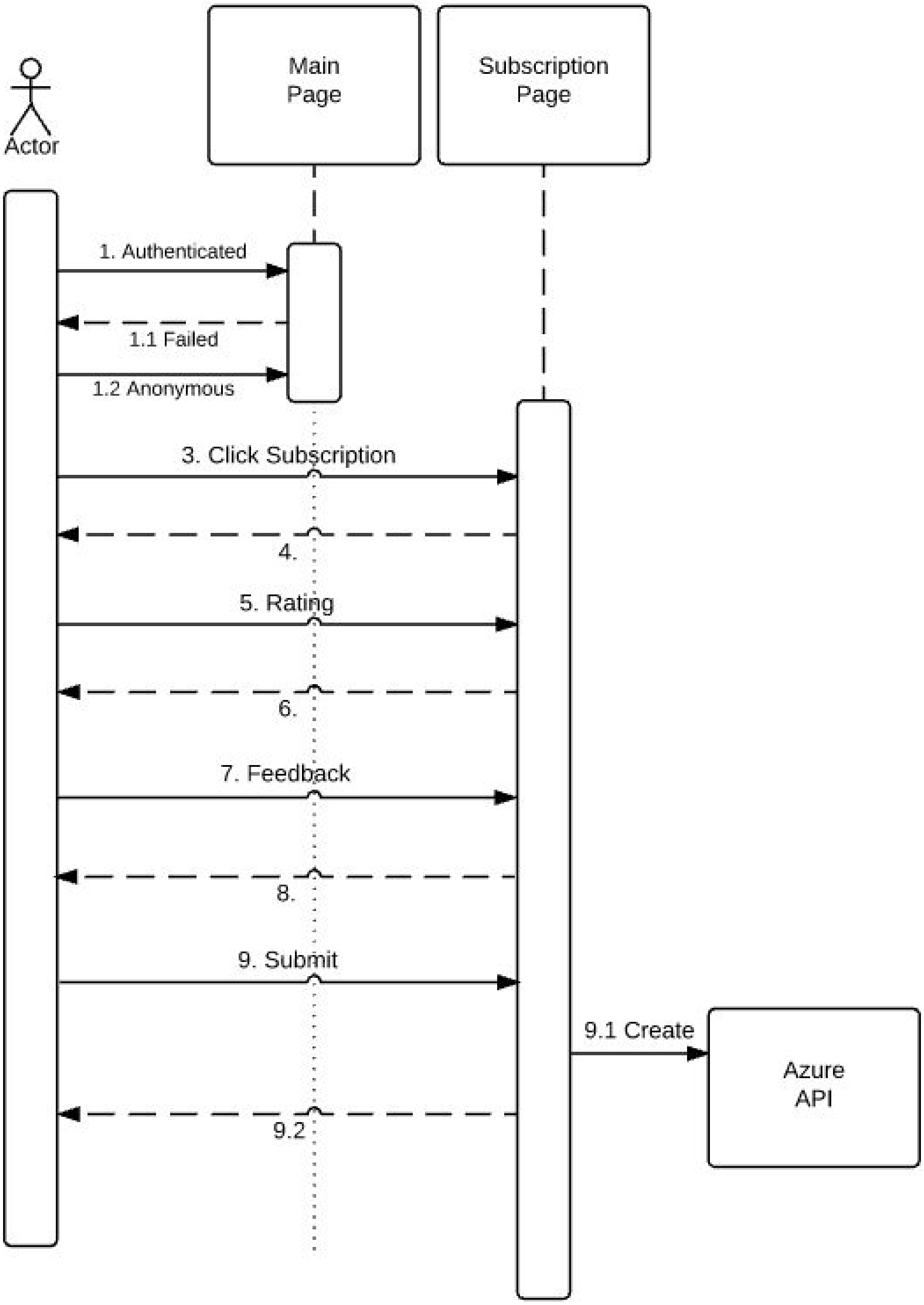
WaitTime	
ID	String
RestaurantID	String
Group	Int
WaitTime	int

Subscription	
ID	String
UserID	String
RestaurantID	String
Rating	Int

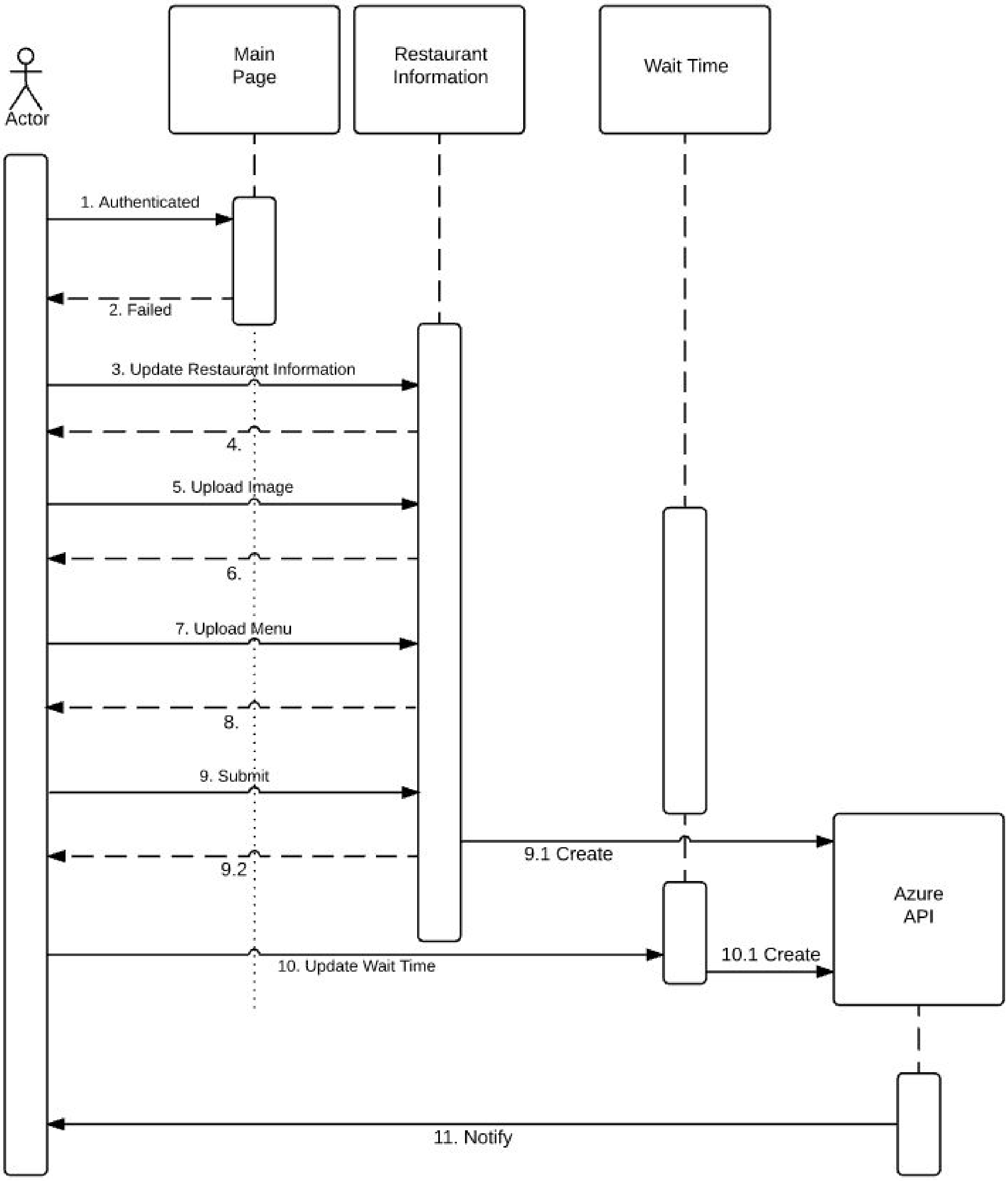
- There will be 4 SQL tables.
- Restaurant table keep data about each restaurant information. It can be join with the other table with the RestaurantID.
- The WaitTime table keep waittime from each restaurant it also separate waittime by number of people in a party (2,4,6,etc people).
- The User table keep track of each user. It has the IdpID from the authentication source. We can also join with UserID
- The Subscription table keep track of user user subscription as well as their feedback to the restaurant.

2.3 Sequence Diagrams

2.3.1 Client User Application



2.3.2 Client Restaurant Application


Actor
Main Page
Restaurant Information
Wait Time
1. Authenticated
2. Failed
3. Update Restaurant Information
4.
5. Upload Image
6.
7. Upload Menu
8.
9. Submit
9.1 Create
9.2
Azure API
10. Update Wait Time
10.1 Create
11. Notify