Name: **SHADA, AKINGBOLADE**

StudentID: **1661269**

CandidateID: **40676**

Customized Restaurants

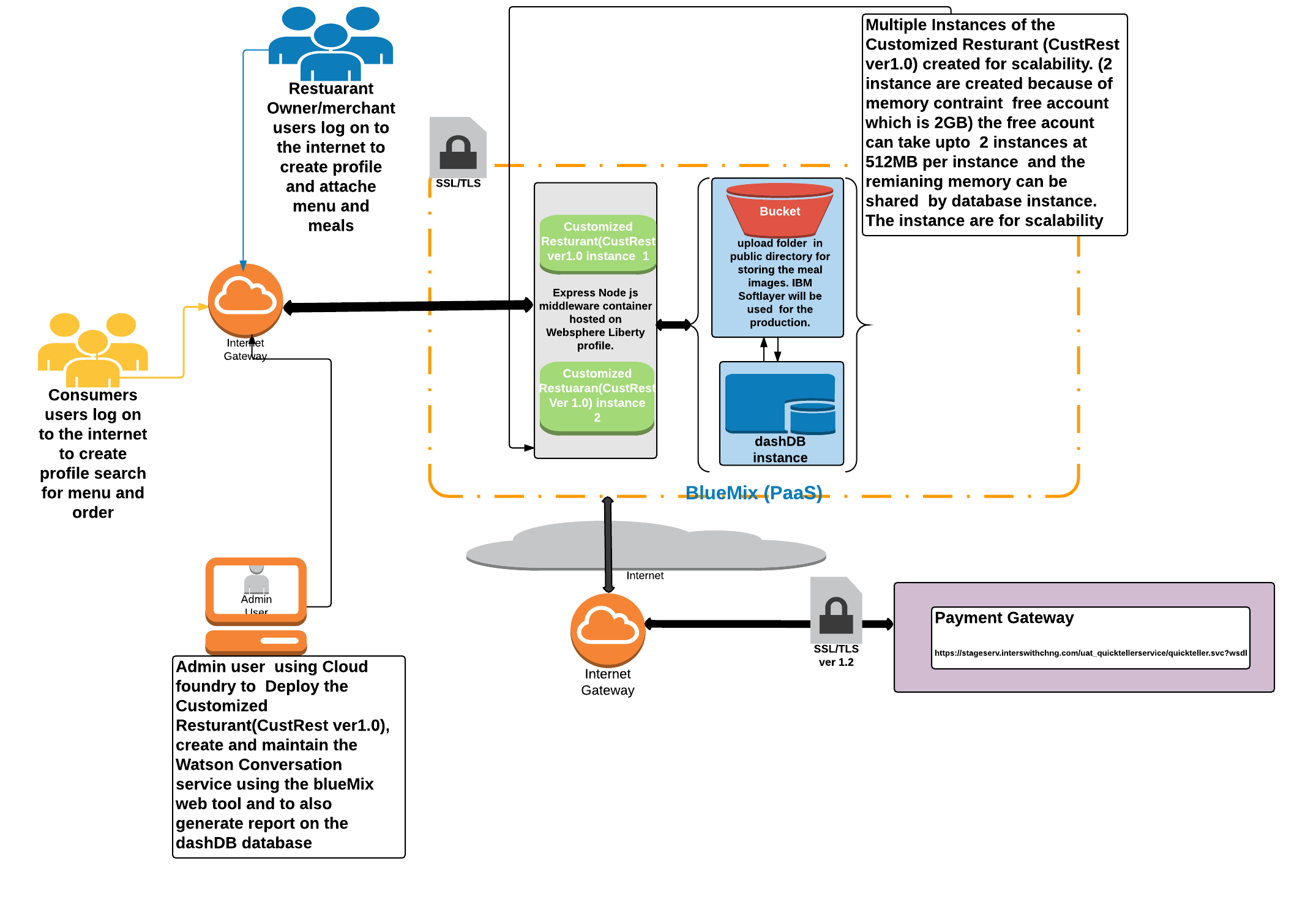
**Overview**

This web application is a brainchild aimed at bringing the cost of food menu to a reasonably and affordable price for third world countries, to boast the information awareness, about the ingredient that is used in making the menu. The web application also addresses the platform that enables the restaurant owners/merchants to register all the meals with respect to their culinary capacity. The restaurant owner setups the menus joined to their profiles. The consumer creates their profile with their addresses and search for meals or menus that have been created on the platform within the threshold distance or setup neighborhood based on the geographical location distance from the prospective consumer and the restaurant merchants. The web application would be deployed on IBM BlueMix when this would be created in Large Scale and the database would also be dashDB, an IBM flagship solution that exploits the brute force of parallelism (MPP)and columnar storage to salvage the ever growing data

challenge. The mysql database is used to create this prototype. The express framework is use to implement the Model View Controller (MVC) architecture and a placebo kind of payment gateway is used to mimic the payment gateway integration for paying for the meals. The port for **https** is **4301** and port for **http** is **4300**

Figure 1.0

**Architecture for Customized Restaurant Web Application**



URL for Restaurant Merchant: <https://localhost:4301/restaurant>

URL for Restaurant Merchant: <http://localhost:4300/restaurant>

URL for Consumer: https://localhost:4301/

URL for Consumer:: http://localhost:4300/

**Web Technologies headings and grades**

**HTML(awarded grade A)**

The view is implemented as a html page rendered by the Express framework JavaScript page. The validation used is also based **on** express validation framework which is installed by running *npm install express-validator* on the administrator profile of the workstation or machine the web application will be deployed. The order at the backend can be downloaded into a pdf with use of *npm install -g html-pdf* installed

**CSS(awarded grade A)**

On mouse-hovering the specific menu, the ingredients and price of the menu is shown. This is achieved by creating a style.css file functions *food\_menu\_hover\_content, single\_food\_item\_down\_content, single\_food\_item\_content\_title* css folder of the *public* directory. I used *icofont icofont-ui-zoom-in* for zomming in and out the meal image because the width of the image is capped to 100 and also the *icofont-shopping-cart*  cart image to select the meal item for ordering.

**JS(awarded grade A)**

The client validation are adapted from the jquery download from [6] and the calendar script used to feed in date while the custome is ordering meals. The price field is also validated to two decimal places and the sum of the orders is implemented as client side javascript *saveDelivery()* in the orders.ejs .

**PNG(awarded grade A)**

The images used in application are the uploaded meals by the restaurant merchants and the upload is ca

**Image rendering**

The uploaded dishes or menus images are capped at maximum of 1024 kilobytes. The file name of the menu image is hashed and stored in a column in the database table column of type VARCHAR (50). This is done to reference the filename of the dish using the html image object. The images file are stored in *upload* folder of the *public* folder, the filenames are hashed to prevent indiscriminate search of menu images belonging to other restaurant merchants/owners in the database, by just changing the filename that is referenced in the URL(user resource locator), that renders the image. The image files are stored in a file media server hosted in folder *upload*in *public*folder. In order to host the web application in IBM Bluemix Cloud, the data will be stored in file storage server which is know as **IBM softlayer** [3] provisioned in the IBM cloud.

*<img src="/../uploads/<%=data[i].IMAGE%>" alt="cart food" width="100">*

**SVP (awarded grade B)**

The are picture used for the Consumer page while searching for menu is download and the cart and zooming images are downloaded [7] and programmatically displayed over the menu image for the user to select and order the menu. This can be seen in index.ejs and order.ejs

**SERVER (awarded grade A)**

**Installed Self signed certificate TLS ver1.2**

**Directory restriction**

The Directory restriction on the site is implemented by creating various script in the router folder to administrate the various object classes for restaurant merchants, Customer and the administrator that is known as user.

**Encryption Standard**

The encryption standard used in saving the password text is the md5 algorithm. The md5 is installed on the server and the directive is call in the app.js the md5 hashing is also used to save the menu image filenames in the upload folder of the public director.

**Database Connection pool**

The connection pool is implemented in the app,js for efficient database management. The dashDB for the BlueMix cloud hosting is used installing the *npm install ibm\_db*

**Express Auto testing**

The validation used is also based **on** express validation framework which is installed by running *npm install express-validator* on the administrator profile of the workstation or machine the web application will be deployed.

**Integration to test payment gateway**

This is also implemented in the app.js but commented out.

Integration with test Payment gateway having SSL and TLS ver1.2 by implementing an override of the HTTPS connection factory of the AXIS2 framework that that wraps, the test payment gateway. *https://stageserv.interswitchng.com/uat\_quicktellerservice/quickteller.svc?wsdl*

The above wsdl url is wrapped into couple of java classes and archived into jar file and this is called by one of the main classes responsible for the specific payment advice. The payment advice is linked to the specific Restaurant merchant’ bank account, for processing. Other data clean ups, transformation and loading of the historical data into the datawarehouse, can also be implemented using this approach, which involves creating a kind of late binding object that invokes connection via the specific channels spanning , HTTPS, HTTP,WSDL, remote database etc.

**Image storage pattern**

Express is a middleware that allows you to specify the file upload size by implementing the following script in the app.js .

app.use(bodyParser.json({limit: '1mb'}));

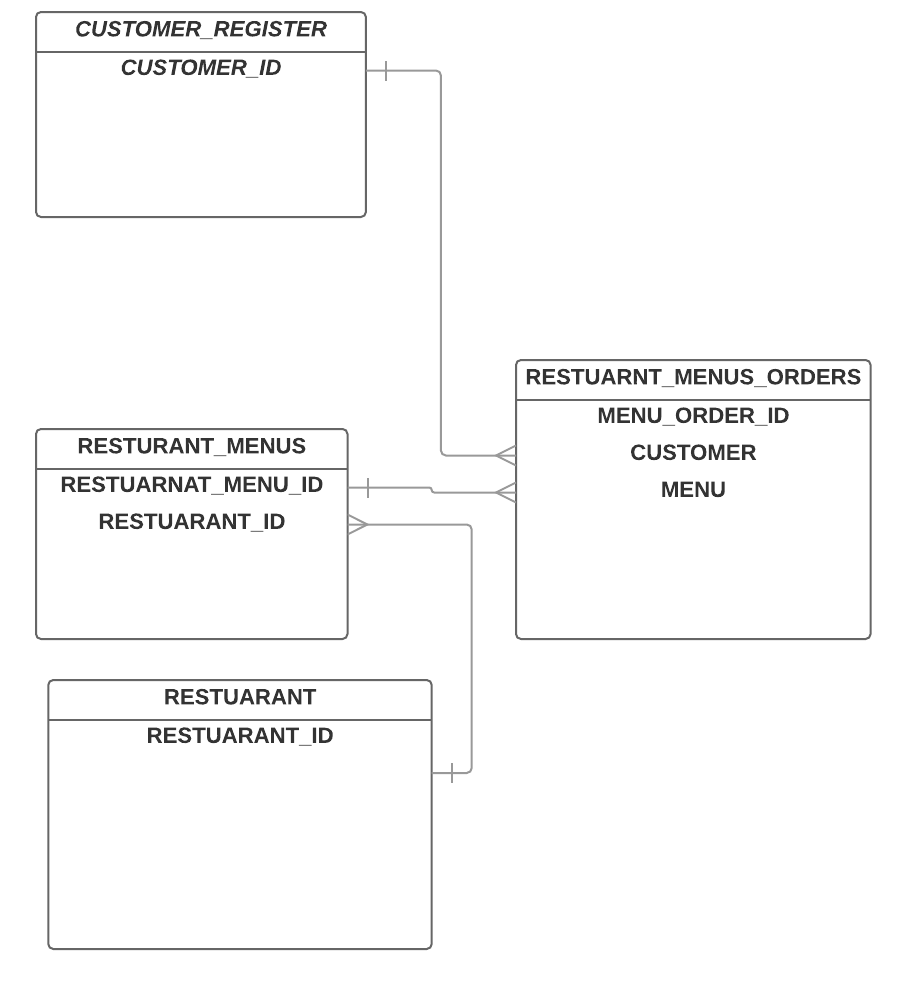
app.use(bodyParser.urlencoded({limit: '1mb', extended: true }));

**DATABASE(awarded grade A)**

**Model**: The part of our application that will deal with the database or any data-related functionality. The application has four tables namely: *CUSTOMER\_REGISTER, RESTAURANT, RESTAURANT\_MENUS, and RESTURANT\_MENU\_ORDER*. The tables are linked to enforce referential integrity based on second normal form of normalization. All query execution to interrogate the data make use of prepare and executed methodology to the militate against the SQL injection. Also the database authentication procedure is implemented by creating a stored procedure in the database to enhance the speed of authentication when more users enroll. The database objects: the tables, view and stored procedures are wrapped around the node js script and linked to the **router** directory. The connection is dropped and released after use. The connection is implemented in the *app.js and the connection pool* is set to 10 for this prototype application*.* The output of the data to be rendered on the express JavaScript [5] page which outputs HTML is done with json payload passed as *data*. The figure 2.0 represents the Entity relationship diagram of the four tables used to create the web application which a second normal form design.

Figure 2.0

**Entity Relation Diagram for the tables**



**Dynamic page (awarded grade A)**

The Express frame created allows me to reuse the headers and footers which reduce the coding efforts.

There two types of headers; the Consumer header are without the admin prefix while the consumer headers are not. The \*.ejs files for headers and footers are located in the layout folder. There is also dynamic script for consumer landing page which is <http://localhost:4300/> . This displays food menus in the database across all Restaurant marchants.

References

1. [*https://www.ibm.com/cloud-computing/bluemix/file-storage*](https://www.ibm.com/cloud-computing/bluemix/file-storage)
2. <https://www.npmjs.com/package/express-validation>
3. <https://knowledgelayer.softlayer.com/topic/file-storage>
4. <https://www.ibm.com/analytics/us/en/technology/cloud-data-services/dashdb/>
5. <https://expressjs.com>
6. <http://jqueryui.com/download/>
7. https://www.iconfinder.com/search/?q=Cart