

Final Report

Project Title: fav.food

Submission Date: 11 Dec 2017

I. Introduction

This is an idea to develop a web application called “*fav.food*” for the people in university to browse and vote for the food selling in the campus. By introducing this service, all the customer within campus, regardless of students, staff or visitors, can benefit from it to know the best restaurant and most popular food within campus area. By browsing this web app, users will know the name of the restaurant, their featured food, the addresses, opening hours and even their average prices. Moreover, in long term development plan, users are also enabled to share with the latest promotion event through our web app so that other users can get informed in time.

II. Background

According to the development plan of our web app. The first phrase of this web app development will focus on a small area and serve a limited group of people. It will be unique service and a good complementary to the other large scale of food review applications such as *Yelp*, *HungryGoWhere* or *OpenRice* [1-4] - it has higher accuracy in serving users and it will be a very lightweight web service. Most importantly, we are quicker in knowing the latest update of the food & beverage business in the campus, and we are confident with providing our user the latest promotions happening throughout the university.

III. Planned Features

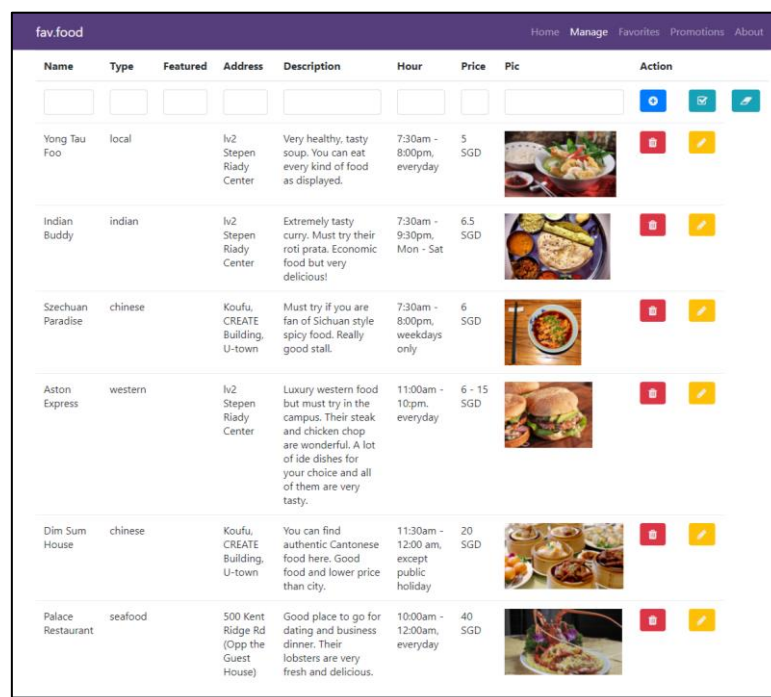
According to the long term development plan, the features of *fav.food* could be summarized as below:

Feature Code	Title	Description
Fo1	Item List	List of food stalls with cover picture.
Fo2	User Module	Identify user by registration/login.
Fo3	Favourites	User can add any item into favourites list.
Fo4	Comment	User can submit his/her comments.
Fo5	Voting	User can like/unlike any item, and the total number of "likes" will be reflected on website.
Fo6	Ranking	List of items will be sorted according to the number of "likes".
Fo7	Promotion	User can create a promotion event on a specific item (similar with Fo4-Comment).

IV. Design & Implementation

As a MEAN full stack project, four key frameworks, MongoDB, Express, AngularJS and Node.js, are used during the implementation [5]. MongoDB will handle all the data modelling, storage and access protocol in the back-end; AngularJS is responsible for connecting the back-end and the front-end, which is also known as the Control role in MVC architecture. The Express and Node.js provides a platform to bind all of the components and frameworks together to realise an application on the web.

The details of the design & implementation of this project is discussed in 4 parts, the back-end (Model of MVC), front-end (View of MVC), communication in the middle (Control of MVC), and last but not least, the mobile platform compatibility.









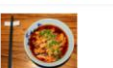











Name	Type	Featured	Address	Description	Hour	Price	Pic	Action
Yong Tau Foo	local		lv2 Stepen Riady Center	Very healthy, tasty soup. You can eat every kind of food as displayed.	7:30am - 8:00pm, everyday	5 SGD		 
Indian Buddy	indian		lv2 Stepen Riady Center	Extremely tasty curry. Must try their roti prata. Economic food but very delicious!	7:30am - 9:30pm, Mon - Sat	6.5 SGD		 
Szechuan Paradise	chinese		Koufu, CREATE Building, U-town	Must try if you are fan of Sichuan style spicy food. Really good stall.	7:30am - 8:00pm, weekdays only	6 SGD		 
Aston Express	western		lv2 Stepen Riady Center	Luxury western food but must try in the campus. Their steak and chicken chop are wonderful. A lot of ide dishes for your choice and all of them are very tasty.	11:00am - 10pm, everyday	6 - 15 SGD		 
Dim Sum House	chinese		Koufu, CREATE Building, U-town	You can find authentic Cantonese food here. Good food and lower price than city.	11:30am - 12:00 am, except public holiday	20 SGD		 
Palace Restaurant	seafood		500 Kent Ridge Rd (Opp the Guest House)	Good place to go for dating and business dinner. Their lobsters are very fresh and delicious.	10:00am - 12:00am, everyday	40 SGD		 

Figure 1 the Manage Page which provides UI to exercise CRUD actions

Back-End – Data Model and REST APIs

The back-end of *fav.food* is presented to user by applying the MongoDB framework. All data entities are stored in database of MongoDB. The access of database and modification of data entity are achieved by using MongoJS library of Node.js.

The data model used in this project is called “restaurant”. All the data is stored in a database named “*restaurantlist*”. The database schema design is presented as below:

```
Model: restaurant
- id: ObjectID
- name: string
- type: string
- addr: string
- description: string
- price: string
- hour: string
- featured: boolean
- pic: URL
```

Here is a sample of restaurant data entity presented in JSON format:

```
{
  "name": "Pizza Hut",
  "type": "fastfood",
  "addr": "2 College Ave West",
  "description": "A global fastfood brand offering many classic types of Pizza",
  "hour": "10:00am - 9:00pm, weekdays only",
  "price": "15 SGD"
}
```

The database could be access and modified through REST APIs [6]. By making use of four kinds of different HTTP protocols, GET, POST, PUT and DELETE, we can achieve all the operations on data in the back-end.

In *fav.food* project, as a web app listing out an exhaustive list of restaurants in a given area, the common operations involved during the implementation could basically summarized as the CRUD (Create, Read, Update and Delete) actions [7].

The details REST API provided for CRUD actions are listed below:

(1) Create (HTTP POST)

We use HTTP Post action to create new data entity into the dataset.

The code snippet supporting this action in *app.js*:

```
app.post('/restaurantlist', function (req, res) {
  console.log(req.body);
  db.restaurantlist.insert(req.body, function(err, doc) {
    res.json(doc);
  });
});
```

This operation could be simulated in Postman by submitting a POST request with input:

```
{
  "name": "Pizza Hut",
  "type": "western",
  "addr": "2 College Ave West",
  "description": "A global fastfood brand offering many classic types of Pizza",
  "hour": "10:00am - 9:00pm, everyday",
  "price": "$15"
}
```

(2) Read (HTTP GET)

We use HTTP Get action to read one or multiple entities from the dataset. There are two routing pattern for this action, one is *restaurantlist/*, which get the list of all existing data, the other one is *restaurantlist/:id* that return the specific data entity by searching its ID.

The code snippet supporting this action in *app.js*:

```
app.get('/restaurantlist', function (req, res) {
  console.log("GET ALL");
  db.restaurantlist.find(function (err, docs) {
    console.log(docs);
    res.json(docs);
  });
});
```

```

    });
  });
  app.get('/restaurantlist/:id', function (req, res) {
    var id = req.params.id;
    console.log(id);
    db.restaurantlist.findOne({_id: mongoose.ObjectId(id)},
      function (err, docs) {
        res.json(docs);
      });
  });
});

```

(3) Update (HTTP PUT)

We use HTTP Put action to modify data entity into the dataset. To update details of a restaurant, we need to specify the ID of a restaurant first.

The code snippet supporting this action in *app.js*:

```

app.put('/restaurantlist/:id', function (req, res) {
  var id = req.params.id;
  console.log(req.body.name);
  db.restaurantlist.findAndModify({
    query: {_id: mongoose.ObjectId(id)},
    update: {$set: {
      name: req.body.name,
      type: req.body.type,
      addr: req.body.addr,
      description: req.body.description,
      pic: req.body.pic,
      hour: req.body.hour,
      price: req.body.price,
      featured: req.body.featured
    }}, new: true}, function(err, doc) {
    res.json(doc);
  });
});

```

Similar with HTTP POST request, this operation could be simulated in Postman by submitting a PUT request with input:

```

{
  "name": "Pizza Hut",
  "type": "fastfood",
  "addr": "2 College Ave West",
  "description": "A global fastfood brand offering many classic types of Pizza",
  "hour": "10:00am - 9:00pm, weekdays only",
  "price": "$15"
}

```

(4) Delete (HTTP DELETE)

We use HTTP Delete action to remove an entity from the dataset. By specifying the ID of data entity, we could achieve the deletion through submitting a HTTP Delete request to *restaurantlist/:id*.

The code snippet supporting this action in *app.js*:

```

app.delete('/restaurantlist/:id', function (req, res) {
  var id = req.params.id;
  console.log(id);
});

```

```

db.restaurantlist.remove({_id: mongoose.ObjectId(id)},
    function(err, doc) {
        res.json(doc);
    });
});

```

Front-End – HTML with Bootstrap Design

The front-end of the web app is written by HTML language. Thanks to the latest *Bootstrap* framework, our website has a clean and beautiful user interface and user experience [8]. By applying Bootstrap grid mechanism, our website can easily adapt with different size of browsers and screens.

In addition to the Bootstrap, we also include the *font-awesome* library to display icons with better user experience, and in the end we apply a Bootstrap-based theme called *item-portfolio* on our website to finalize our website design [9, 10].

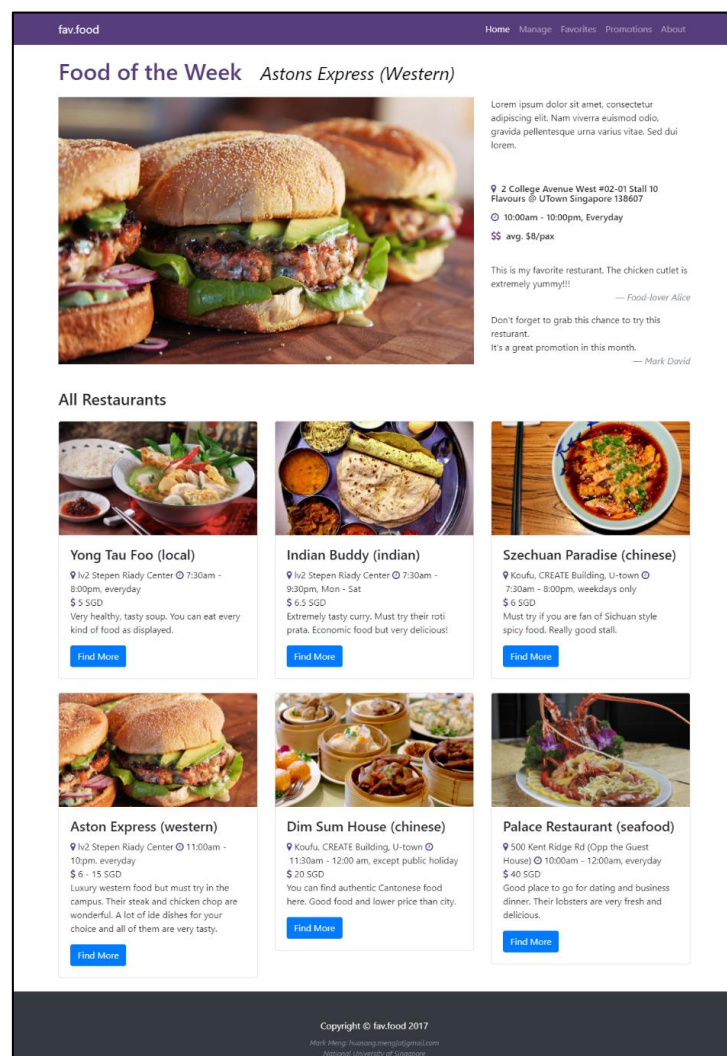


Figure 2 the Index Page of fav.food web app

Here we provides code snippets for displaying featured restaurant in *index.html*, which contain the usage of the theme defined CSS, Bootstrap grid and font-awesome icons:

```

<!-- Featured Item Row -->
<div class="row">
  <div class="col-md-8">
    
    </div>

    <div class="col-md-4">
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam viverra euismod
        odio, gravida pellentesque urna varius vitae. Sed dui lorem.</p>
      <br>

      <h6 class="my-3"><i class="fa fa-map-marker" aria-hidden="true"
style="color:#563d7c"></i>&nbsp;
        2 College Avenue West
        #02-01 Stall 10 Flavours @ UTown
        Singapore 138607
      </h6>

      <h6 class="my-3">
        <i class="fa fa-clock-o" aria-hidden="true" style="color:#563d7c"></i>
        &nbsp;10:00am - 10:00pm, Everyday
      </h6>

      <h6 class="my-3">
        <i class="fa fa-usd" aria-hidden="true" style="color:#563d7c"></i>
        <i class="fa fa-usd" aria-hidden="true" style="color:#563d7c;
          margin-left:-5px;">
        </i>
        &nbsp;&nbsp; avg. $8/pax
      </h6>
      <br>

      <blockquote class="blockquote">
        <p class="small mb-0">
          This is my favorite resturant. The chicken cutlet is extremely yummy!!!
        </p>
        <footer class="blockquote-footer text-right"><em>Food-lover Alice</em>
      </blockquote>

      <blockquote class="blockquote">
        <p class="small mb-0">Don't forget to grab this chance to try this resturant.
          <br>It's a great promotion in this month.</p>
        <footer class="blockquote-footer text-right"><em>Mark David</em>
      </blockquote>
    </div>
  </div>
</div>

```

Control in-the-middle Design & Implementation

The *AngularJS* is selected as the control mechanism of our project MVC architecture. Under the help of AngularJS, we could easily convert the data model into the front-end HTML components.

The communication control source code is implemented in a JavaScript file named *controller.js*. Here I provide some code snippets for the data CRUD handling:

```

...
$scope.addRestaurant = function() {

```

```

    console.log($scope.restaurant);
    $http.post('/restaurantlist',
        $scope.restaurant).success(function (res) {
        console.log(res);
        refresh();
    });
};

$scope.removeRestaurant = function(id) {
    console.log(id);
    $http.delete('/restaurantlist/' + id).success(
        function (res) {
            refresh();
        });
};

$scope.update = function() {
    console.log($scope.restaurant._id);
    $http.put('/restaurantlist/' + $scope.restaurant._id,
        $scope.restaurant).success(
        function (res) {
            refresh();
        });
};
...

```

```

{ name: 'PizzaHut',
  type: 'western',
  addr: '2 College Ave West',
  description: 'Fast food Pizza',
  hour: '7:30am - 8:00pm, everyday',
  price: '15 SGD',
  pic: 'http://static5.businessinsider.com/image/53908351ecad04ca746ba577-480/pizza-hut-cmo-sp.jpg' }

```

Figure 3 the log output of the creation of a restaurant in database

The usage of AngularJS in front-end HTML in our project contains ng-app, ng-controller, ng-repeat, ng-model and ng-click. For example, the code snippet of displaying full list of restaurants in *index.html* is like:

```

<div class="row">
  <div class="col-md-4 col-sm-6 mb-4"
    ng-repeat="restaurant in restaurantlist">
    <div class="card">
      
      <div class="card-body">
        <h4 class="card-title">
          {{restaurant.name}} ({{restaurant.type}})
        </h4>
        <p class="card-text">
          <i class="fa fa-map-marker"
            aria-hidden="true"
            style="color:#563d7c">
          </i>&nbsp;{{restaurant.addr}}
          <i class="fa fa-clock-o"
            aria-hidden="true"
            style="color:#563d7c">
          </i>&nbsp;{{restaurant.hour}}</br>
          <i class="fa fa-usd"
            aria-hidden="true"
            style="color:#563d7c">
          </i>&nbsp;{{restaurant.price}}</br>
          {{restaurant.description}}
        </p>
      </div>
    </div>
  </div>
</div>

```

```
</div>
</div>
</div>
</div>
```

Mobile Platform Compatibility – Cordova

Cordova is an *Apache*-family solution to transform a traditional website into a mobile platform adaptive application [11]. By using *Cordova*, our web app could be installed as a standalone application on multiple mainstream mobile platform such as iOS, Android, Windows Phone and BlackBerry OS, without writing duplicated source code.

Our *fav.food* project also benefits from *Cordova* framework to become friendly to mobile user. There are some screenshots taken on the iOS simulator which could demonstrate the usage of our web app on smaller size screen on mobile devices.

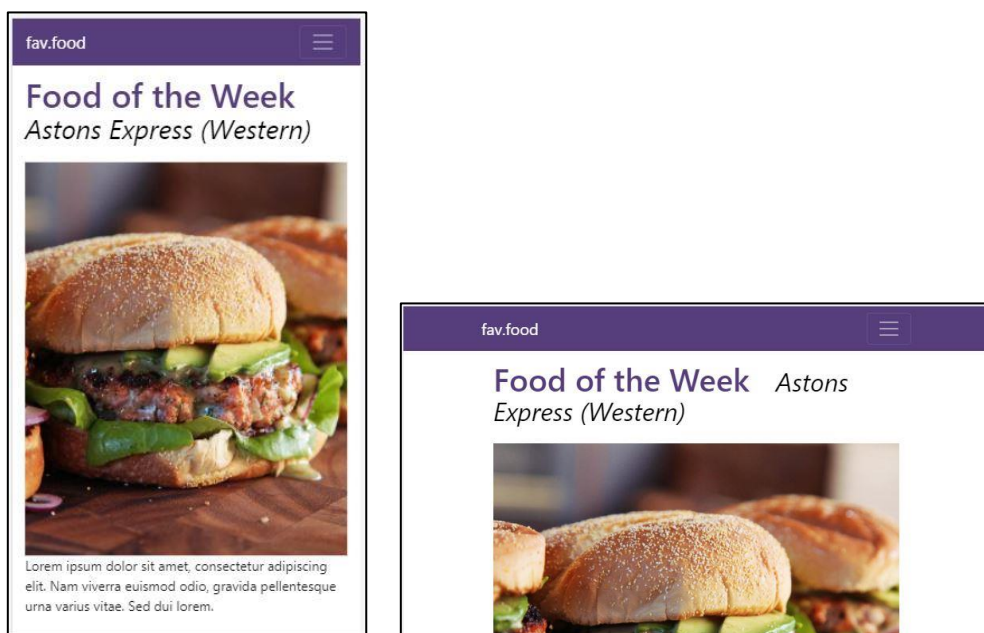


Figure 4 the screenshots of *fav.food* web app on Cordova emulation (left: horizontal screen, right: portrait screen)

V. Conclusions

This report introduces the application design and software structure of the project named "*fav.food*". Throughout this two-month period of Capstone project course, we have successfully accomplished the first stage of *fav.food* project development.

During this project, the REST API is designed during early stage of the development and later is adopted in MongoDB integration. In order to facilitate the website usage on various platform, the Bootstrap framework and other relevant resources are used in front-end design to enhance the UI/UX of the web app. For the connection between front-end HTML and back-end data models, the AngularJS framework is selected to handle data access and modification. Last but not least, a mobile platform optimization tool called *Cordova* is applied in our *fav.food* project to make it friendly to the mobile users.

This may not be the end of our project. In the short future, we hope to further the integration of all key component of MEAN full stack development, including replacing static data storage by MongoDB, implementation of user profiling and applying Cordova for better user experience on mobile platform. We believe that we can make fully use of the knowledge and experience we learnt throughout the journey of this Coursera specialization in the industry and our career in the future.

References

- [1] Staff, “*Made-in-Singapore: 12 Singaporean food apps to whet your appetite*”, <https://www.stuff.tv/sg/features/made-in-singapore-12-singaporean-foodapps-whet-your-appetite>
- [2] Yelp, “*Singapore Restaurants, Dentists, Bars, Beauty Salons, Doctors – Yelp*”, <https://www.yelp.com.sg/singapore>
- [3] HungryGoWhere, “*Singapore Food Guide, Restaurant Reviews & Reservation – HungryGoWhereI*”, <http://www.hungrygowhere.com/>
- [4] OpenRice, “*Singapore Restaurants Guide Singapore Restaurant | OpenRice Singapore*”, <https://sg.openrice.com/en/singapore>
- [5] Mean.io, “*Mongo Express Angular Node*”, <http://mean.io/>
- [6] Wikipedia, “*Representational State Transfer*”, https://en.wikipedia.org/wiki/Representational_state_transfer
- [7] Wikipedia, “*Create, Read, Update and Delete*”, https://en.wikipedia.org/wiki/Create,_read,_update_and_delete
- [8] Bootstrap, “*Bootstrap, a sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development*”, <https://getbootstrap.com>
- [9] Font-Awesome, “*Font-Awesome Icons – The complete set of 675 icons in Font Awesome 4.7.0*”, <http://fontawesome.io/icons/>
- [10] Start Bootstrap Theme, “*Portfolio Item*”, <https://startbootstrap.com/template-overviews/portfolio-item/>
- [11] Cordova, “*Apache Cordova*”, <https://cordova.apache.org/>

END OF REPORT