**Exercise03\_03\_01 – Step 1**

quoteManager.js

updateQuotes () {

------------------------------------

quoteManager.js

updateQuotes() {

return new Promise((resolve, reject) => {

});

------------------------------------

quoteManager.js

module.exports = {

updateQuotes() {

return new Promise((resolve, reject) => {

api.getAllPizzas(function (err, pizzas) {

var newData = [], pizza;

for (var key in pizzas) {

pizza = pizzas[key];

newData.push({

ticker: pizza.ticker,

nextQuote: pizza.getNext()

});

}

console.log(`${JSON.stringify(newData)}

updating quotes`);

callback(null, newData);

});

});

}

};

------------------------------------

quoteManager.js

console.log(`${JSON.stringify(newData)}

updating quotes`);

resolve(newData);

quoteManager.js

var newData = [],

pizza;

if (err) {

reject(err);

}

------------------------------------

quoteManager.js

if (err) {

reject(err);

}

else {

for (var key in pizzas) {

pizza = pizzas[key];

newData.push({

ticker: pizza.ticker,

nextQuote: pizza.getNext()

});

}

console.log(`${JSON.stringify(newData)}

updating quote`);

resolve(newData);

}

**Exercise03\_03\_01 – Step 2**

market.js

quoteManager.updateQuotes()

function (err, newData) {

socket.emit('new\_data', JSON.stringify(newData));

});

------------------------------------

market.js

quoteManager.updateQuotes()

.then((newData) => {

socket.emit('new\_data', JSON.stringify(newData));

});

------------------------------------

market.js

.then((newData) => {

socket.emit('new\_data', JSON.stringify(newData));

})

.catch((err) => {

console.error(err);

});

**Exercise03\_03\_01 – Step 3**

api.js

function getPizza(ticker) {

return new Promise((resolve, reject) => {

});

------------------------------------

api.js

return new Promise((resolve, reject) => {

request('http://localhost:' + localPort + '/pizza/' +   
 ticker, function (error, res, body) {

if (callback) {

callback(error,   
 Pizza.hydrate(JSON.parse(body)));

}

});

});

------------------------------------

api.js

request('http://localhost:' + localPort + '/pizza/' +   
 ticker, function (error, res, body) {

if (error) {

reject(error);

}

else {

resolve(Pizza.hydrate(JSON.parse(body)));

}

});

------------------------------------

popGen.js

function getNewestSlice(callback) {

api.getPizza('HAWA')

.then((pizza) => {

if (callback) {

callback(null, { ticker: 'HAWA', quote:   
 pizza.getLast() });

}

});

}

popGen.js

.then((pizza) => {

if (callback) {

callback(null, { ticker: 'HAWA', quote:   
 pizza.getLast() });

}

})

.catch((err) => {

callback(err);

});

**Exercise03\_03\_01 – Step 4**

main.js

module.exports = function (request, reply) {

var context = {};

const promises = [

popGen.getPopularSlices(),

popGen.getMostPopular(),

popGen.getNewestSlice(),

popGen.getMostImproved(),

dataStore.getPizzas()

];

------------------------------------

main.js

Promise.all(promises)

.then((results) => {

});

main.js

Promise.all(promises)

.then((results) => {

const context = {};

return reply.view('index', context);

});

------------------------------------

main.js

.then((results) => {

const context = {

popSlices: results[0],

mostPopular: results[1],

newestSlice: results[2],

mostImproved: results[3],

pizzas: results[4]

};

------------------------------------

main.js

})

.catch((err) => {

console.error(err);

});

popGen.js

function getPopularSlices() {

return new Promise((resolve, reject) => {

\_getFinalQuotes()

.then((finalQuotes) => {

const orderedQuotes = \_.orderBy(finalQuotes,   
 ['quote'], ['desc']);

resolve(\_.take(orderedQuotes, 4));

})

.catch(reject);

});

}

------------------------------------

popGen.js

function getMostPopular () {

return new Promise((resolve, reject) => {

\_getFinalQuotes()

.then((finalQuotes) => {

const mostPopular =   
 finalQuotes.reduce(function (best, curr) {

if (curr.quote > best.quote) {

return curr;

}

return best;

}, { quote: 0 });

resolve(mostPopular);

})

.catch(reject);

});

}

popGen.js

function getNewestSlice() {

return new Promise((resolve, reject) => {

api.getPizza('HAWA')

.then((pizza) => {

resolve({

ticker: 'HAWA',

quote: pizza.getLast()

});

})

.catch((err) => {

reject(err);

});

});

}

------------------------------------

popGen.js

function getMostImproved () {

return new Promise((resolve, reject) => {

api.getAllQuotes()

.then((allQuotes) => {

const diffQuotes = [];

for (const key in allQuotes) {

diffQuotes.push({

ticker: key,

diff: allQuotes[key][allQuotes[key].length - 1] -   
 allQuotes[key][0],

quote: allQuotes[key][allQuotes[key].length - 1]

});

}

const mostImproved = diffQuotes.reduce(function (best,   
 curr) {

if (curr.diff > best.diff) {

return curr;

}

return best;

}, { diff: 0});

resolve(mostImproved);

})

.catch(reject);

});

}

api.js

function getAllQuotes() {

return new Promise((resolve, reject) => {

request('http://localhost:' + localPort + '/quotes',   
 function (error, res, body) {

if (error) {

reject(error);

} else {

resolve(JSON.parse(body));

}

});

});

}

------------------------------------

api.js

function getAllPizzas() {

return new Promise((resolve, reject) => {

request('http://localhost:' + localPort + '/pizzas',   
 function (error, res, body) {

if (error) {

reject(error);

} else {

const staticPizzas = JSON.parse(body),

pizzas = [];

for (var ix in staticPizzas) {

pizzas.push  
 (Pizza.hydrate(staticPizzas[ix]));

}

resolve(pizzas);

}

});

});

}

------------------------------------

datastore.js

function getAllQuotes() {

return new Promise((resolve) => {

resolve(data.quotes);

});

}

datastore.js

function getPizzas() {

return new Promise((resolve) => {

resolve(data.pizzas);

});

}

------------------------------------

quotes.js

module.exports = function (request, reply) {

dataStore.getAllQuotes()

.then(reply)

.catch(reply);

};

------------------------------------

quoteManager.js

module.exports = {

updateQuotes() {

return new Promise((resolve, reject) => {

api.getAllPizzas()

.then((pizzas) => {

const newData = [];

let pizza;

for (const key in pizzas) {

pizza = pizzas[key];

newData.push({

ticker: pizza.ticker,

nextQuote: pizza.getNext()

});

}

console.log(`${JSON.stringify(newData)}   
 updating quotes`);

resolve(newData);

})

.catch(reject);

});

}

};

popGen.js

function \_getFinalQuotes () {

return new Promise((resolve, reject) => {

api.getAllQuotes()

.then((allQuotes) => {

const finalQuotes = [];

for (const key in allQuotes) {

finalQuotes.push({

ticker: key,

quote: allQuotes[key][allQuotes[key].length - 1],

diffLast:   
 \_percentOf(allQuotes[key][allQuotes[key]  
 .length - 2],   
 allQuotes[key][allQuotes[key].length - 1])

});

}

resolve(finalQuotes);

})

.catch(reject);

});

}

------------------------------------

src/handlers/pizza.js

dataStore.getPizza(ticker)

.then(reply)

.catch(reply)

};

**Exercise03\_03\_01 – Step 5**

popGen.js

const finalQuotes = new Set();

------------------------------------

popGen.js

for (const key in allQuotes) {

finalQuotes.add({

------------------------------------

popGen.js

const mostPopular = [...finalQuotes].reduce(function   
 (best, curr) {

------------------------------------

popGen.js

const orderedQuotes = \_.orderBy([...finalQuotes],   
 ['quote'], ['desc']);

**Exercise03\_03\_01 – Step 6**

dataStore.js

const data = new Map();

------------------------------------

dataStore.js

data.set('quotes', require('../mock/quotes'));

------------------------------------

dataStore.js

data.set('pizzas', pizzas);

------------------------------------

dataStore.js

return data.get('quotes')[ticker];

------------------------------------

dataStore.js

callback(null, data.get('quotes'));

------------------------------------

dataStore.js

callback(null, data.get('pizzas'));

------------------------------------

dataStore.js

callback(null, data.get('pizzas')[ticker]);

------------------------------------

dataStore.js

realPizzas[pizza[0]] = new Pizza(startingDate,   
 data.get('quotes')[pizza[0]], ...pizza);