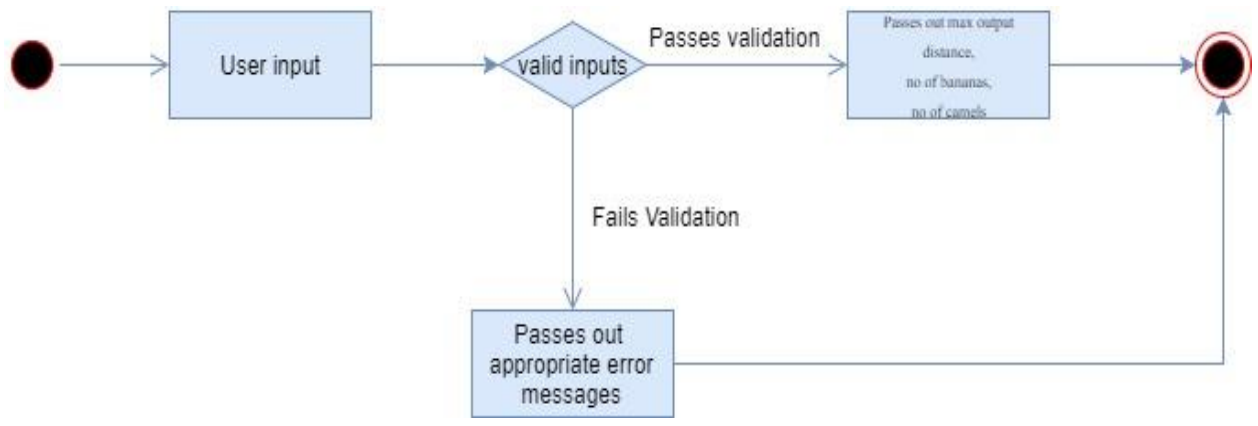
Cotta & Cush Software Engineering Challenge Document

Date	**Version**	**Author**
5-06-2019	1.0	Oyegoke Oluwatoyosi

Table of Contents

Design Overview

Below is a UML diagram which describes the design overflow of the solution



User Manual

The solution requires a user to supply two inputs:

1. The Number of Camels
2. The total number of bananas

These input parameters must be positive integers greater than or equals to 1 as the application would reject parameters which don't conform to this rule.

The number of camels can either be 1 or 3 based on the two scenarios.

The total number of bananas bust not be less than 1 and not greater than 3000 and the application rejects inputs that violates this rule.

To get submit the inputs, click on the "Get Output" button and it will display the total inputted number of bananas, total inputted number of camels, the output left to be sold in the market and the distance in km.

Code Documentation

File List

- App.js
- data/data.js
- function/function.js
- node_modules
- public/css/style.css
- routes/camel.js
- test/tools.js
- views/index.ejs
- package.json
- package-lock.json

App.js: This is the entry point of the application. It is where the server is started, routes/controller and dependencies are being are being required and utilized.

It also specifies the default path for the applications index page and redirects all other paths to it i.e

```
// require routes
const camelRoute = require('./routes/camel')

// use route
app.use("/camel",camelRoute)

// catch all route redirects to camel route
app.get('*', (req, res)=>{
    return res.redirect("/camel")
})
```

To Start the application, run the command “node app.js” in the projects root directory from the terminal.

Code	Use
const express = require("express");	Requires the express framework
const app = express();	Initiates express and saves it in a variable app
app.use(express.json());	Express middleware allowing use of json data

app.use(express.urlencoded({ extended: true }));	Express middleware allowing use of nested objects passed through the body
app.set("view engine", "ejs")	Sets the templating engine to ejs
app.use(express.static(__dirname+'/public'))	Serving the static folder which contains our css file
const camelRoute = require('./routes/camel')	Requires the controller file
app.listen(port, ()=>console.log(`Server is listening on ${port}`))	Tells the browser to start the app on a specified port

Data/data.js : This is the file where the “data” class is being defined. This is a class of the output passed out based on the user input. It contains method which are returned when a set of user inputs are valid and when user inputs are invalid. The Data class is being exported and used in the routes/camel.js file.

Function/function.js: This file contains the main function used in the controller file. The function accepts two parameters i.e n = number of camels and b = number of babnanas and uses them to determine the expected maximum output and total distance in km. The function is exported and used in the routes/camel.js file.

Node_modules: This folder contains all installed packages used for the project.

Public/css/style.css: This file contains the styling of the rendered html page.

Routes/camel.js : This is the main controller file. It has two http actions namely get and post. The get route renders the index.ejs file to the browser and passes the data initiated from the Data class as a variable named “response”.

The post action accepts the data from the form in the req.body object. Within this block of code, user inputs are being validated and after passing the validation are passed to the mainfunction required from the function/function.js file. Upon receiving the output, it modifies the response array sent to the index file and redirects back to the previous page.

Test/tools.js : This file contains the unit testing logic. It specifies the input parameters for the main function and its expected output. To check if the test is passed or failed, run the command “mocha” from the command line in the projects root directory.

Views/index.ejs : This is the html file rendered unto the browser. It uses a templating engine called “ejs” which helps us write javascript and html together

Package.json : This is the a file that contains the list of dependencies and dev-dependencies used on the project. To install all dependencies, run the command “npm install”.