Andela

The Andela Developer Challenge

Build A Product: **Book-A-Meal**

Andela

OVERVIEW

What?

Book-A-Meal is an application that allows customers to make food orders and helps the food vendor know what the customers want to eat.

How?

This project is broken down into challenges and completion of all challenges would contribute greatly to your learning towards becoming a full-stack developer. Upon completion, you would have built a world-class full-stack (front-end and back-end) Javascript application.

Why?

Andela distributes opportunities. We disseminate Learning and catalyse Technology leadership. The project is founded on the premise that aspiring Technology Leaders learn programming whilst building things that matter and that the best way to learn is by building a complete product.

This project has one objective: *create opportunities for learning where they build real products. In this way, we inspire change in the African tech landscape.*

Andela

BUILD A PRODUCT: Meal Booking App

Required Features

1. Users can create an account and log in
2. Admin (Caterer) should be able to manage (i.e: add, modify and delete) meal options in the application. Examples of meal options are: Beef with rice, Beef with fries etc
3. Admin (Caterer) should be able to setup menu for a specific day by selecting from the meal options available on the system.
4. Authenticated users (customers) should be able to see the menu for a specific day and select an option out of the menu.
5. Authenticated users (customers) should be able to change their meal choice.
6. Admin (Caterer) should be able to see the orders made by the user
7. Admin should be able to see amount of money made by end of day

Extra Features

1. Authenticated users (customers) should be able to see their order history
2. Authenticated users (customers) should be able to get notifications when the menu for the day has been set.
3. Admin (Caterer) should be able to see order history
4. The application should be able to host more than one caterer.

Andela

Challenge 2: Setup Server-Side & Create API endpoints

Timelines

- **Expected Length to Complete: 1 week**
- **Due Date: 27th of April, 2018**

Challenge Summary

You are expected to create a set of API endpoints already defined below and use javascript data structures to store data in memory (don't use a database).

NB:

- *Ensure that Challenge 1 is completed and merged to the **develop** branch before you get started.*

Tools

- Server side Framework: **Node.js/Express**
- Linting Library: **Eslint**
- Style Guide: **Airbnb**
- Testing Framework: **Mocha**

Guidelines

1. Setup the **server side of the application** using the **specified** framework
2. Setup linting library and ensure your work follows **Airbnb style guide** requirements.
3. Setup test framework
4. Version your API using url versioning starting, with the letter "v". A simple ordinal number would be appropriate and avoid dot notation such as 2.5. An example of this will be : https://somewebapp.com/api/v1/users
5. Using separate branches for each feature, create version 1 (v1) of your **RESTful API** to power front-end pages
6. At minimum, you should have the following API endpoints working:

EndPoint	Functionality
`GET /meals/`	Get all the meal options
`POST /meals/`	Add a meal option
PUT /meals/<mealId>	Update the information of a meal option
DELETE /meals/<mealId>	Remove a meal option
`POST /menu/`	Setup the menu for the day

Andela

GET /menu/	Get the menu for the day
POST /orders	Select the meal option from the menu
PUT /orders/orderId	Modify an order
GET /orders	Get all the orders

7. Write tests for the functions and API endpoints
8. Ensure to test all endpoints and see that they work using Postman.
9. Integrate **TravisCI** for Continuous Integration in your repository (with *ReadMe* badge).
10. Integrate **test coverage reporting** (e.g. **Coveralls**) with badge in the *ReadMe*.
11. Obtain **CI badges** (e.g. from **Code Climate** and **Coveralls**) and add to *ReadMe*.
12. Improve your codebase based on the test coverage report and CI issues identified.
13. Request your LFA to review the pull request above
14. Ensure the app gets hosted on Heroku.

Helpful Links

- Use the recommended Git Workflow, Git branch, Commit Message and Pull Request (PR) standards.
- All Javascript **MUST** be written in **ES6 or higher** and should be transpiled to **ES5** using **Babel** or similar tools.
- Adhere strictly to the Airbnb style guide for **ES6**.
- Use **OOP**. Structure your code using **ES6 classes** and **methods**
- Use the appropriate data structure to hold data and manipulate data
- Classes/modules **MUST** respect the ***SRP*** (Single Responsibility Principle) and **MUST** use the ***ES6*** methods of *module* ***imports and exports.***

  NB*: A good place to start would be ensuring that eslint has been setup and is working!*
- Use **SASS/SCSS** to implement all custom styling.
- Before you begin this section, ensure to review these materials
  - Guide to Restful API design
  - Best Practices for a pragmatic RESTful API

Targeted Skills

After completing this challenge, you should have learnt and be able to demonstrate the following skills:

Andela

ES6 + Babel	All Javascript **MUST** be written in **ES6 or higher** and should use **Babel** to transpile down to **ES5**
OOP + SRP	Classes/modules **MUST** respect the ***SRP*** (Single Responsibility Principle) and **MUST** use the ***ES6*** methods of *module* ***imports and exports.***

.eslint - Airbnb style guides	Use a .**eslin**t in the your root directory of your project as your eslint configuration (in your IDE) to expose Javascript syntax errors / nitpicks. Make sure to ***extend the airbnb styleguide*****.**
Continuous Integration	Integrate **HoundCI** for style checking commits in your PRs. Integrate a CI tool (e.g. **TravisCI**) to also run tests and report pass/fail state with badge in readme and also test coverage reporting (e.g. Coveralls) with badge in the readme. Obtain CI badges from **Code Climate** and **Coveralls**. *These should be in the readme*

NodeJS	You were required to create a server directory in your repository and project directory in challenge 1. The“server” directory would contain the server-side implementation you come up with in this challenge required to power the front-end. Setup the back-end (server side) of the application with **NodeJS - Express**.
Data Structures	Implement non-persistent data storage using arrays and JSON objects.

Build API	Download and install the Google Chrome app **Postman.** This would be used to test the API you are building.

Self / Peer Assessment Guidelines

Use this as general guidelines to assess quality of your work. Peers, mentors, and facilitators should use this to give **feedback** on areas that should be improved on.

Criterion	Does not Meet Expectation	Meets Expectations	Exceed Expectations
Programming Logic	Fails to write a function that returns a value	Translates requirements into working functions which are implemented with	Optimizes code to effectively use system resources

Andela

		best practices in mind	
Data Structures	Fails to implement CRUD or Implements CRUD with persistence	Implements CRUD without persistence	Uses the most optimal data structure for each operation
Test-Driven Development	Solution did not attempt to use TDD	Writes tests that pass and achieves 70% test coverage	Writes tests that pass and achieves 100% code coverage
Object-Oriented Programming	Fails to write classes that incorporate basics of OOP	Writes classes with attributes and methods	Reuses code via class inheritance
HTTP & Web API	Fails to develop an API that meets the requirements specified	Successfully develops an api that gives access to all the specified end points	Handles a wide array of HTTP error codes and the error messages are specific
Continuous Integration	Fails to integrate all required CI tools.	Successfully integrates all tools with relevant badges added to ReadMe.	