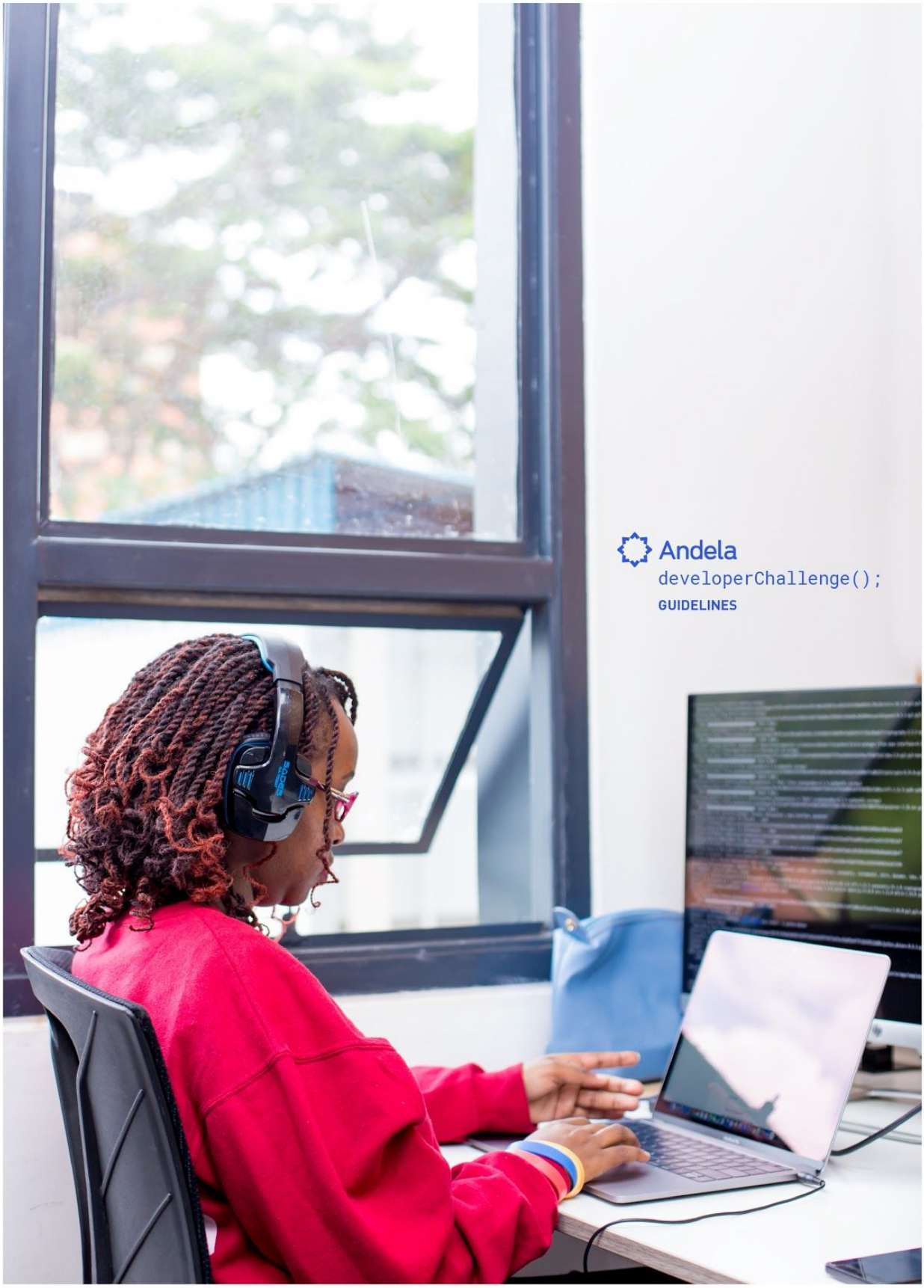
Andela



Andela

developerChallenge();

GUIDELINES

© Andela Confidential

Andela

Andela Developer Challenge

Build A Product: **Auto Mart**

© Andela Confidential

Andela

BUILD A PRODUCT: Auto Mart

Project Overview

Auto Mart is an online marketplace for automobiles of diverse makes, model or body type. With Auto Mart, users can sell their cars or buy from trusted dealerships or private sellers.

Project Timelines

- **Total Duration**: **7 weeks**
- **Final Due Date**: **June 21st, 2019**

Required Features

1. User can sign up.
2. User can sign in.
3. User (seller) can post a car sale advertisement.
4. User (buyer) can make a purchase order.
5. User (buyer) can update the price of his/her purchase order.
6. User (seller) can mark his/her posted AD as sold.
7. User (seller) can update the price of his/her posted AD.
8. User can view a specific car.
9. User can view all unsold cars.
10. User can view all unsold cars within a price range.
11. Admin can delete a posted AD record.
12. Admin can view all posted ads whether sold or unsold.

Optional Features

- User can reset password.
- User can view all cars of a specific body type.
- User can add multiple pictures to a posted ad.
- User can **flag/report** a posted AD as fraudulent.
- User can view all unsold cars of a specific make (manufacturer).
- User can view all used unsold cars.
- User can view all new unsold cars.

© Andela Confidential

Andela

Preparation Guidelines

These are the steps you ought to take to get ready to start building the project

Steps

1. Create a **Pivotal Tracker Project**
   ***Tip**: find how to create a Pivotal Tracker Project here.*

2. Create a **Github Repository, add a README, and clone it to your computer**
   ***Tip**: find how to create a Github Repository here.*

© Andela Confidential

Andela

Challenge 1 - Create UI Templates

Challenge Summary

You are required to create UI templates with **HTML**, **CSS**, and **Javascript**.

Timelines

- **Duration:** **2 Weeks**
- **Due Date:** **May 17th, 2019**

NB:

- *You are not implementing the core functionality yet, you are only building the User Interface (UI) elements, pages, and views!*
- *You are to create a pull request for each feature in the challenge and then merge into your develop branch.*
- *Do not use any CSS framework e.g Bootstrap, Materialize, sass/scss.*
- *Do not use any JavaScript frontend framework e.g React, Angular, Vue.*
- *Do not download or use an already built website template.*

Guidelines

1. **On Pivotal Tracker, create user stories to setup the User Interface (UI) elements:**
    a. A page/pages where a **user** (client) can do the following:
        i. sign up.
        ii. sign In.
        iii. post a car sale advertisement.
        iv. update the price of his/her posted car sale ad.
        v. make a purchase order.
        vi. update the price or his/her purchase order.
        vii. mark his/her posted AD as sold.
        viii. **flag/report** a posted AD as fraudulent.
        ix. view a specific car.
        x. view all unsold cars, with the option to filter the cars using **any** or all of these attributes (price range, make (manufacturer), car state (new or used))

    b. A page/pages where an **admin** can do the following:
        i. View all cars, whether sold or unsold.

© Andela Confidential

ii. Delete a posted AD.

c. **On Pivotal Tracker, create stories to capture any other tasks not captured above. A task can be feature, bug or chore for this challenge.**

2. **On a feature branch, create a directory called UI in your local Git repo and build out all the necessary pages specified above and UI elements that will allow the application function into the UI directory.**
3. **Host your UI templates on GitHub Pages.**

Tip: *It is recommended that you create a* ***gh-pages*** *branch off the branch containing your UI template. When following the GitHub Pages guide, select* ***"Project site" >> "Start from scratch"****. Remember to choose the* ***gh-pages*** *branch as the* ***source*** *when configuring Repository Settings. See a detailed guide* ***here****.*

Target skills

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

Skill	Description	Helpful Links
Project management	Using a project management tool (Pivotal Tracker) to manage your progress while working on tasks.	• To get started with Pivotal Tracker, use Pivotal Tracker quick start. • Here is a sample template for creating Pivotal Tracker user stories.
Version control with GIT	Using GIT to manage and track changes in your project.	• Use the recommended git **branch**, **pull request** and **commit message** naming convention. • Use the recommended Git Workflow, Commit Message and Pull Request (PR) standards.
Front-End Development	Using HTML and CSS to create user interfaces.	• See this tutorial • See this tutorial also
UI/UX	Creating good UI interface and user experience	• See rules for good UI design here • See this article for More guide • For color palettes, see this link

© Andela Confidential

Andela

Self / Peer Assessment Guidelines

Use this as general guidelines to assess the 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
Project management	Fails to break down modules into smaller, manageable tasks. Cannot tell the difference between chores, bugs, and features	Breaks down each module into smaller tasks and classifies them. Constantly updates the tool with progress or lack of it	Accurately, assigns points to the tasks. Informs stakeholders of project progress/blockers in a timely manner
Version Control with Git	Does not utilize branching but commits to master branch directly instead.	Utilizes branching, pull-requests, and merges to the develop branch. Use of recommended commit messages.	Adheres to recommended GIT workflow and uses badges.
Front-End Development	Fails to develop the specified web pages using **HTML/CSS/JavaScript** or uses an already built out website template, or output fails to observe valid **HTML/CSS/Javascript** syntax or structure.	Successfully develops web pages while observing standards such as doctype declaration, proper document structure, no inline CSS in HTML elements, and HTML document has consistent markup	Writes modular CSS that can be reused through markup selectors such as class, id. Understands the concepts and can confidently rearrange divs on request.
UI/UX	The page is unresponsive, elements are not proportional, the color scheme is not complementary and uses alerts to display user feedback	The page is responsive (at least across mobile, tablet and desktops), the color scheme is complementary, and uses properly designed dialog boxes to give the user feedback	User interface is well thought out, resulting in a memorable user experience. UI is functional with captivating aesthetics.

© Andela Confidential

Andela

Challenge 2: Create API endpoints

Challenge Summary

You are expected to create a set of API endpoints defined in the API Endpoints Specification section and use data structures to store data in memory (don't use a database).

Timelines

- **Duration:** **3 weeks**
- **Due Date:** **June 7th, 2019**

NB:

- *You are to create a pull request for each feature in the challenge and then merge into your develop branch*

Tools

- Server side Framework: **Node**/**Express**
- Linting Library: ***ESLint***
- Style Guide: ***Airbnb***
- Testing Framework: ***Mocha*** *or* ***Jasmine***

Guidelines

1. **Setup linting library and ensure that your work follows the specified style guide requirements.**
2. **Setup the test framework.**
3. **Write unit-tests for all API endpoints.**
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 1.0. An example of this will be: https://somewebapp.com/api/v1/.**
5. **Using separate branches for each feature, create version 1 (v1) of your RESTful API to power front-end pages.**
6. **On Pivotal Tracker, create user stories to setting up and testing API endpoints that do the following using data structures:**
    - User can sign up.
    - User can sign in.

© Andela Confidential

- User (seller) can post a car sale advertisement.
- User (buyer) can make a purchase order.
- User (buyer) can update the price of his/her purchase order.
- User (seller) can mark his/her posted AD as sold.
- User (seller) can update the price of his/her posted AD.
- User can view a specific car.
- User can view all unsold cars.
- User can view all unsold cars within a price range.
- Admin can delete a posted AD record.
- Admin can view all posted ads whether sold or unsold.

Optional Features

- User can reset password.
- User can add multiple pictures to a posted ad.
- User can view all cars of a specific body type.
- User can view all used unsold cars.
- User can view all new unsold cars.
- User can **flag/report** a posted AD as fraudulent.
- User can view all unsold cards of a specific make (manufacturer).

7. **Use cloudinary to store your images and only save the URL in your application's database.**
8. **Use API Blueprint, Slate, Apiary or Swagger to document your API. Docs should be accessible via your application's URL.**
9. **On Pivotal Tracker create stories to capture any other tasks not captured above. The tasks can be feature, bug or chore for this challenge.**
10. **Setup the server side of the application using the specified framework**
11. **Ensure to test all endpoints and see that they work using Postman.**
12. **Integrate TravisCI for Continuous Integration in your repository (with *ReadMe* badge).**
13. **Integrate test coverage reporting (e.g. Coveralls) with a badge in the *ReadMe*.**
14. **Obtain CI badges (e.g. from Code Climate and Coveralls) and add to *ReadMe*.**
15. **Ensure the app gets hosted on Heroku.**

© Andela Confidential

16. **At the minimum, you should have the API endpoints listed under the API endpoints specification on page 14 working. Also refer to the entity specification on page 13 for the minimum fields that must be present in your data models.**

API Response Specification

The API endpoints should respond with a JSON object specifying the HTTP ***status*** code, and either a ***data*** property (on success) or an ***error*** property (on failure). When present, the ***data*** property is always an ***object*** *or an* ***array***.

On Success

```
{
  "status" : 200,
  "data" : {...}
}
```

On Error

```
{
  "status" : 404,
  "error" : "relevant-error-message"
}
```

The status codes above are provided as samples, and by no way specify that all success reponses should have **200** or all error responses should have **400.**

Target skills

After completing this challenge, you should have learned and able to demonstrate the following skills.

Skill	Description	Helpful Links
Project management	Using a project management tool (Pivotal Tracker) to manage your progress while working on tasks.	• To get started with Pivotal Tracker, use Pivotal Tracker quick start. • Here is a sample template for creating Pivotal Tracker user stories.
Version control with GIT	Using GIT to manage and track changes in your project.	• Use the recommended git **branch**, **pull request** and **commit message** naming convention. • Use the recommended Git Workflow, Commit Message and Pull Request (PR) standards.

© Andela Confidential

Andela

Cloudinary with Node.js	How to use cloudinary Node.js SDK	• https://cloudinary.com/documentation/node_integration • https://itnext.io/file-uploading-using-cloudinary-complete-workflow-abfab093fdab
HTTP & Web services	Creating API endpoints that will be consumed using Postman	• Guide to Restful API design • Best Practices for a pragmatic RESTful API
Test-driven development	Writing tests for functions or features.	• Getting started with TDD in Node Node.Js
Data structures	Implement non-persistent data storage using data structures.	• JavaScript data types and data structures.
Continuous Integration	Using tools that automate build and testing when the code is committed to a version control system.	• Setup Continuous Integration with Travis CI in Your Nodejs App
Holistic Thinking and big-picture thinking	An understanding of the project goals and how it affects end users before starting on the project.	• Node-postgres • Node.js postgresql tutorial

Self / Peer Assessment Guidelines

Use this as general guidelines to assess the 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
Project management	Fails to break down modules into smaller, manageable tasks. Cannot tell the difference between chores, bugs, and features	Breaks down each module into smaller tasks and classifies them. Constantly updates the tool with progress or lack of it	Accurately, assigns points to the tasks. Informs stakeholders of project progress/blockers in a timely manner
Version Control with Git	Does not utilize branching but commits to master branch directly instead.	Utilizes branching, pull-requests, and merges to the develop branch. Use of recommended commit	Adheres to recommended GITflow workflow and uses badges.

© Andela Confidential

		messages.	
Programming logic	The code does not work in accordance with the ideas in the problem definition.	The code meets all the requirements listed in the problem definition.	The code handles more cases than specified in the problem definition. The code also incorporates best practices and optimizations.
Test-Driven development	Unable to write tests. The solution did not attempt to use TDD	Writes tests with 60% test coverage.	Writes tests with coverage greater than 60%.
HTTP & Web Services	Fails to develop an API that meets the requirements specified	Successfully develops an API that gives access to all the specified endpoints	Handles a wide array of HTTP error code and the error messages are specific.
Data Structures	Fails to implement CRUD or Implements CRUD with persistence	Implements CRUD without persistence	Uses the most optimal data structure for each operation
Continuous Integration • Travis CI • Coverall	Fails to integrate all required CI tools.	Successfully integrates all tools with relevant badges added to ReadMe.	

© Andela Confidential

Andela

Entity Specification

Users

```
{
  “id” : Integer,
  “email” : String,
  “first_name” : String,
  “last_name” : String,
  “password” : String,
  “address” : String,
  “is_admin” : Boolean,
   ...
}
```

Cars

```
{
  “id” : Integer,
  “owner” : Integer,                // user id
  “created_on” : DateTime,
  “state” : String,                 // new,used
  “status” : String,                // sold,available - default is available
  “price” : Float,
  “manufacturer” : String,
  “model” : String,
  “body_type” : String,             // car, truck, trailer, van, etc
   ...
}
```

Order

```
{
  “id” : Integer,
  “buyer” : Integer,                // user id
  “car_id” : Integer,
  “amount” : Float,                 // price offered
  “status” : String,                // [pending, accepted, or rejected]
   ...
}
```

Flags

```
{
  “id” : Integer,
  “car_id” : Integer,
  “created_on” : DateTime,
  “reason” : String,                // [pricing, weird demands, etc]
  “description” : String,
   ...
}
```

© Andela Confidential

Andela

API Endpoint Specification

Endpoint: POST /auth/signup
Create user account

Response spec:

```
{
  “status” : Integer,
  “data” : {
    “token” :  “45erkjherht45495783”
    “id”: Integer,      // id of newly created user
    “first_name”: String,
    “last_name”: String,
    “email”: String,
    ...
  }
}
```

Endpoint: POST /auth/signin
Login a user

Response spec:

```
{
  “status” : Integer,
  “data” : {
    “token” :  “45erkjherht45495783”
    “id”: Integer,      // user id
    “first_name”: String,
    “last_name”: String,
    “email”: String,
    ...
  }
}
```

Endpoint: POST /car/
Create a car sale ad.

Response spec:

```
{
  “status” : Integer,
  “data” : {
    “id” : Integer,
    “email” : String,
    “created_on” : DateTime,
    “manufacturer” : String,
    “model” : String,
```

© Andela Confidential

```
    "price" : Float,
    "state" : String,
    "status" : String,
    ...
  }
}
```

Endpoint: POST /order/

Create a purchase order.

Response spec:

```
}
  "status" : Integer,
  "data" : }
    "id" : Integer,
    "car_id" : Integer,
    "created_on" : DateTime,
    "status" : String,
    "price" : Float,
    "price_offered" : Float,
    ...
  }
}
```

Endpoint: PATCH /order/<:order-id>/price

Update the price of a purchase order. A user can only update the price of his/her purchase order while the order's status still reads **pending.**

Response spec:

```
}
  "status" : Integer,
  "data" : }
    "id" : Integer,
    "car_id" : Integer,
    "status" : String,
    "old_price_offered" : Float,
    "new_price_offered" : Float,
    ...
  }
}
```

© Andela Confidential

Andela

Endpoint: PATCH /car/<:car-id>/status

Mark a posted car Ad as sold.

Response spec:

```
{
  “status” : Integer,
  “data” : {
    “id” : Integer,
    “email” : String,
    “created_on” : DateTime,
    “manufacturer” : String,
    “model” : String,
    “price” : Float,
    “state” : String,
    “status” : String,
    ...
  }
}
```

Endpoint: PATCH /car/<:car-id>/price

Update the price of a car.

Response spec:

```
{
  “status” : Integer,
  “data” : {
    “id” : Integer,
    “email” : String,
    “created_on” : DateTime,
    “manufacturer” : String,
    “model” : String,
    “price” : Float,
    “state” : String,
    “status” : String,
    ...
  }
}
```

Endpoint: GET /car/<:car-id>/

View a specific car.

Response spec:

```
{
  “id” : Integer,
  “owner” : Integer,
  “created_on” : DateTime,
  “state” : String,
```

© Andela Confidential

```
  “status” : String,
  “price” : Float,
  “manufacturer” : String,
  “model” : String,
  “body_type” : String,
   ...
}
```

Endpoint: GET /car?status=available

View all unsold cars.

Response spec:

```
{
  “status” : Integer,
  “data” : [
     {
       “id” : Integer,
       “owner” : Integer,
       “created_on” : DateTime,
       “state” : String,
       “status” : String,
       “price” : Float,
       “manufacturer” : String,
       “model” : String,
       “body_type” : String,
        ...
     },
     {
       “id” : Integer,
       “owner” : Integer,
       “created_on” : DateTime,
       “state” : String,
       “status” : String,
       “price” : Float,
       “manufacturer” : String,
       “model” : String,
       “body_type” : String,
        ...
     }
  ]
}
```

© Andela Confidential

Andela

Endpoint: GET /car?status=available&min_price=**XXXValue**&max_price=**XXXValue**

User can view all unsold cars within a price range..

Response spec:

```
{
  "status" : Integer,
  "data" : [
     {
       "id" : Integer,
       "owner" : Integer,
       "created_on" : DateTime,
       "state" : String,
       "status" : String,
       "price" : Float,
       "manufacturer" : String,
       "model" : String,
       "body_type" : String,
        ...
     },
     {
       "id" : Integer,
       "owner" : Integer,
       "created_on" : DateTime,
       "state" : String,
       "status" : String,
       "price" : Float,
       "manufacturer" : String,
       "model" : String,
       "body_type" : String,
        ...
     }
  ]
}
```

Endpoint: DELETE /car/<:car_id>/

Delete a specific car Ad.

Response spec:

```
{
  "status" : Integer,
  "data" :   "Car Ad successfully deleted"
}
```

© Andela Confidential

Andela

Endpoint: GET /car/

View all posted ads whether sold or available.

Response spec:

```
{
  “status” : Integer,
  “data” : [
     {
       “id” : Integer,
       “owner” : Integer,
       “created_on” : DateTime,
       “state” : String,
       “status” : String,
       “price” : Float,
       “manufacturer” : String,
       “model” : String,
       “body_type” : String,
        ...
     },
     {
       “id” : Integer,
       “owner” : Integer,
       “created_on” : DateTime,
       “state” : String,
       “status” : String,
       “price” : Float,
       “manufacturer” : String,
       “model” : String,
       “body_type” : String,
        ...
     }
  ]
}
```

© Andela Confidential

Andela

OPTIONAL

Endpoint: POST /flag/

flag/report a posted AD as fraudulent.

Response spec:

```
{
  “id” : Integer,
  “car_id” : Integer,
  “reason” : String,
  “description” : String,
}
```

Endpoint: GET /car?status=available&state=new

View all unsold cars of a specific make (manufacturer).

Response spec:

```
{
  “status” : Integer,
  “data” : [
     {
       “id” : Integer,
       “owner” : Integer,
       “created_on” : DateTime,
       “state” : String,
       “status” : String,
       “price” : Float,
       “manufacturer” : String,
       “model” : String,
       “body_type” : String,
        ...
     },
     {
       “id” : Integer,
       “owner” : Integer,
       “created_on” : DateTime,
       “state” : String,
       “status” : String,
       “price” : Float,
       “manufacturer” : String,
       “model” : String,
       “body_type” : String,
        ...
     }
  ]
}
```

© Andela Confidential

Andela

Endpoint: GET /car?status=available&state=used

View all unsold cars of a specific make (manufacturer).

Response spec:

```
{
  “status” : Integer,
  “data” : [
     {
       “id” : Integer,
       “owner” : Integer,
       “created_on” : DateTime,
       “state” : String,
       “status” : String,
       “price” : Float,
       “manufacturer” : String,
       “model” : String,
       “body_type” : String,
        ...
     },
     {
       “id” : Integer,
       “owner” : Integer,
       “created_on” : DateTime,
       “state” : String,
       “status” : String,
       “price” : Float,
       “manufacturer” : String,
       “model” : String,
       “body_type” : String,
        ...
     }
  ]
}
```

Endpoint: GET /car?status=available&manufacturer=**XXXValue**

View all unsold cars of a specific make (manufacturer).

Response spec:

```
{
  “status” : Integer,
  “data” : [
     {
       “id” : Integer,
       “owner” : Integer,
       “created_on” : DateTime,
       “state” : String,
       “status” : String,
       “price” : Float,
       “manufacturer” : String,
       “model” : String,
       “body_type” : String,
        ...
```

© Andela Confidential

```
    },
    {
      "id" : Integer,
      "owner" : Integer,
      "created_on" : DateTime,
      "state" : String,
      "status" : String,
      "price" : Float,
      "manufacturer" : String,
      "model" : String,
      "body_type" : String,
       ...
    }
  ]
}
```

Endpoint: GET /car?body_type=**bodyType**

View all cars of a specific body type.

Response spec:

```
{
  "status" : Integer,
  "data" : [
    {
      "id" : Integer,
      "owner" : Integer,
      "created_on" : DateTime,
      "state" : String,
      "status" : String,
      "price" : Float,
      "manufacturer" : String,
      "model" : String,
      "body_type" : String,
       ...
    },
    {
      "id" : Integer,
      "owner" : Integer,
      "created_on" : DateTime,
      "state" : String,
      "status" : String,
      "price" : Float,
      "manufacturer" : String,
      "model" : String,
      "body_type" : String,
       ...
    }
  ]
}
```

© Andela Confidential