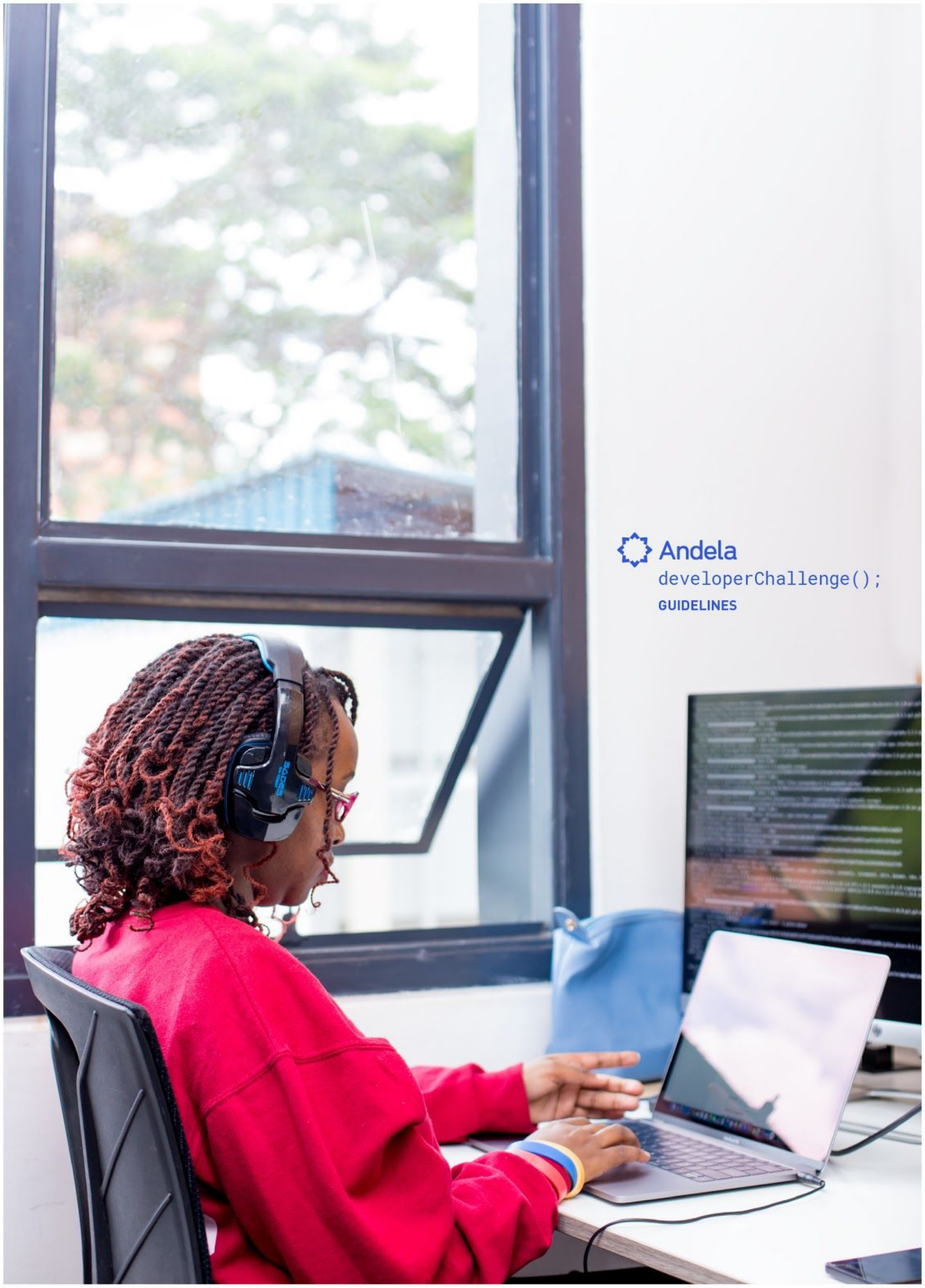
Andela



Andela developerChallenge(); GUIDELINES

© Andela Confidential

Andela

Andela Developer Challenge

Build A Product: **SendIT**

© Andela Confidential

Andela

BUILD A PRODUCT: SendIT

Project Overview

SendIT is a courier service that helps users deliver parcels to different destinations.

Project Timelines

- **Total Duration:** **3 weeks**
- **Final Due Date**: **Tuesday, 27th November, 2018**

Required Features

1. Users can create an account and log in.
2. Users can create a parcel delivery order.
3. Users can change the destination of a parcel delivery order.
4. Users can cancel a parcel delivery order.
5. Users can see the details of a delivery order.
6. Admin can change the **status** and **present location** of a parcel delivery order.
7. The application should display a Google Map with Markers showing the **pickup location** and the **destination**.

Optional Features

1. The application should display a Google Map with a line connecting both Markers (**pickup location** and the **destination**).
2. The application should display a Google Map with computed travel distance and journey duration between the **pickup location** and the **destination**.
3. The user gets real-time email notification when Admin changes the **status** of their parcel or when Admin changes the **present location** of their parcel.

Note:

1. The user can only cancel or change the **destination** of a parcel delivery when the parcel's **status** is yet to be marked as **delivered**.
2. Only the user who created the parcel delivery order can cancel the order.

© Andela Confidential

Preparation Guidelines

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

Steps

1. Create a **Pivotal Tracker Board**

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 a RESTful API

Challenge Summary

You are expected to create all the endpoints required to meet all the requirements listed under the **required features** section and ensure that you persist data with a database**.** You are to write SQL queries that will help you save to, and read from your database. The endpoints are to be secured with JWT where necessary.

Timelines

- **Duration:** **2 Weeks**
- **Due Date:** **Tuesday, 20th November, 2018**

NB:

- *You are to create a pull request for each feature in the challenge and then merge into your develop branch.*
- *All JavaScript **MUST** be written in **ES6 or higher** and should use **Babel** to transpile down to **ES5***
- *Classes/modules **MUST** respect the **SRP** (Single Responsibility Principle) and **MUST** use the **ES6** methods of module **imports and exports.***
- *Do not to use any ORMs for database interactions.*

Tools

- Server side Framework: <*Node.js*>
- 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. **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**
4. **Using separate branches for each feature, create version 1 (v1) of your RESTful API to power front-end pages**
5. **On Pivotal Tracker, create a chore for setting up the database.**

© Andela Confidential

6. **On Pivotal Tracker, create user stories for setting up and testing API endpoints that do the following using database:**
    a. Create a parcel delivery order
    b. Get all parcel delivery orders
    c. Get a specific parcel delivery order
    d. Cancel a parcel delivery order
    e. The user can create user accounts and can sign in to the app.
    f. The user can change the destination of a parcel delivery order.
    g. The user can view all parcel delivery orders he/she has created.
    h. Admin can view all parcel delivery orders in the application.
    i. Admin can change the status of a parcel delivery order.
    j. Admin can change the present location of a parcel delivery order
7. **On Pivotal Tracker, create the story(s) for the implementation of token-based authentication using JSON web token (JWT) and the security of all routes using JSON web token (JWT).**
8. **On Pivotal Tracker, create stories to capture any other tasks not captured above. The tasks could be feature, bug or chore for this challenge.**
9. **On Pivotal Tracker, create user story(s) to implement one or all out of these optional features:**
    a. The user gets real-time email notification when Admin changes the **status** or the **present location** of a parcel. Refer to Optional Features on **page 3.**
    NB: Executing the above optional feature in addition to the minimum set of API endpoints means you have exceeded expectations.
10. **Setup a PostgreSQL database.**
11. **Write tests for all the endpoints.**
12. **Ensure to test all endpoints and see that they work using Postman.**
13. **Use API Blueprint, Slate, Apiary or Swagger to document your API. Docs should be accessible via your application's URL.**
14. **Integrate TravisCI for Continuous Integration in your repository (with *ReadMe* badge).**
15. **Integrate test coverage reporting (e.g. Coveralls) with a badge in the *ReadMe.***
16. **Obtain CI badges (e.g. from Code Climate and Coveralls) and add to *ReadMe*.**
17. **Ensure the app gets hosted on Heroku.**

© Andela Confidential

18. **At a minimum, you should have the API endpoints listed under the API endpoints specification on page 11 working. Also refer to the entity specification on page 10 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 ***array***, even if there's none, one, or several items within it.

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 Workflow, Commit Message and Pull Request (PR) standards.

HTTP & Web services	Creating API endpoints that will be consumed using Postman	• Guide to Restful API design • Best Practices for a pragmatic RESTful API • Web services
Test-driven development	Writing tests for functions or features.	
Continuous Integration	Using tools that automate build and testing when the code is committed to a version control system.	
Databases	Using database to store data	• Node-postgres • Node.js postgresql tutorial
Holistic Thinking and big-picture thinking	An understanding of the project goals and how it affects end users before starting on the project	

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 recommended GIT workflow and uses badges.

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 test coverage greater than 60%.
HTTP & Web Services	Fails to develop an API that meets the requirements specified	Successfully develops an API that performs required tasks over the specified endpoints	All API endpoints provide the appropriate HTTP status codes, headers and messages
Databases	Unable to create database models for the given project	Has a database of normalized tables with relationships. Can store, update and retrieve records using SQL	
Token-Based Authentication	Does not use Token-Based authentication	Makes appropriate use of Token-Based authentication and secures all private endpoints	
Security	Fails to implement authentication and authorization in given project	Successfully implements authentication and authorization in the project	Creates custom and descriptive error messages
Continuous Integration ● Travis CI ● Coveralls	Fails to integrate all required CI tools.	Successfully integrates all tools with relevant badges added to ReadMe.	

Andela

Entity Specification

User

```
{
  “id” : Integer,
  “firstname” : String,
  “lastname” : String,
  “othernames” : String,
  “email” : String,
  “username” : String,
  “registered” : Date,
  “isAdmin” : Boolean,
   ...
}
```

Parcel

```
{
  “id” : Integer,
  “placedBy” : Integer,         // represents the sender
  “weight” : Float,
  “weightmetric” : String,      // [kg]
  “sentOn” : Date,
  “deliveredOn” : Date,
  “status” : String,            // [placed, transiting, delivered]
  “from” : String,              // an address
  “to” : String,                // an address
  “currentLocation” : String,   // an address
   ...
}
```

© Andela Confidential

API Endpoint Specification

Endpoint: GET /parcels
Fetch all parcel delivery orders.

Response spec:

```
{
  “status” : Integer,
  “data” : [{...}, {...}, {...}]
}
```

Endpoint: GET /parcels/<parcelId>
Fetch a specific delivery order.

Response spec:

```
{
  “status” : Integer,
  “data” : [{...}]
}
```

Endpoint: GET /users/<userId>/parcels
Fetch all parcel delivery order by a specific user.

Response spec:

```
{
  “status” : Integer,
  “data” : [{...}, {...}, {...}]
}
```

Endpoint: PATCH /parcels/<parcelId>/cancel
Cancel a specific parcel delivery order.

Response spec:

```
{
  “status” : Integer,
  “data” : [{
     “id” : Integer,
     “message” : “order canceled”
  }]
}
```

Endpoint: POST /parcels
Create a parcel delivery order..

Response spec:

```
{
  “status” : Integer,
  “data” : [{
    “id” : Integer,
    “message” :  “order created”
  }]
}
```

Endpoint: POST /auth/signup
Create a user account.

Response spec:

```
{
  “status” : Integer,
  “data” : [{
    “token” :  “45erkjherht45495783”,
    “user”: {....} // the user object
  }]
}
```

Endpoint: POST /auth/login
Login a user

Response spec:

```
{
  “status” : 200,
  “data” : [{
    “token” : “ahd64jfhHG7832KFM5”,
    “user”: {....} // the user object
  }]
}
```

Endpoint: PATCH /parcels/<parcelId>/destination
Change the destination of a specific parcel delivery order. Only the user who created the parcel should be able to change the destination of the parcel. A parcel’s destination can only be changed if it is yet to be delivered.

Response spec:

```
{
  “status” : Integer,
  “data” : [{
    “id” : Integer,   // the Parcel
    “to”: “updated location address”,
    “message”: “Parcel destination updated”
  }]
}
```

Endpoint: PATCH /parcels/<parcelId>/status

Change the status of a specific parcel delivery order. Only the Admin is allowed to access this endpoint.

Response spec:

```
{
  “status” : Integer,
  “data” : [{
    “id” : Integer,   // the Parcel
    “status”: “the updated status”,
    “message”: “Parcel status updated”
  }]
}
```

Endpoint: PATCH /parcels/<parcelId>/currentlocation

Change the present location of a specific parcel delivery order. Only the Admin is allowed to access this endpoint..

Response spec:

```
{
  “status” : Integer,
  “data” : [{
    “id” : Integer,   // the Parcel
    “currentLocation” : “update location address”,
    “message” : “Parcel location updated”
  }]
}
```

© Andela Confidential

Andela

Challenge 2 - Implement the Front-End App

Challenge Summary

You are expected to create a front-end UI with **HTML, CSS** and **JavaScript.** This UI should be function with and consume data from the API built in **Challenge 1.**
Note that this challenge requires that you implement your front-end logic using standard language capabilities (e.g **ES6** for Javascript**)**.

Timelines

- **Duration:** 1 Week
- **Due Date:** Tuesday, 27th November, 2018

Note:

- *Ensure that **Challenge 1 is completed** and merged to the **develop** branch of your github repository before you get started with this challenge*
- *You are to make use of the native browser **FETCH API** for making requests to the backend API built in Challenge 1*
- *Do **NOT** download or use any already built website template*
- *Do **NOT** use any CSS libraries/framework e.g Bootstrap or Materialize*
- *Do **NOT** use frameworks or libraries like JQuery, Angular, Vue or React*

Guidelines

1. **On Pivotal Tracker, create user stories to set up the User Interface elements:**
   a. User sign-up and sign-in pages.
   b. A page/pages where a user can do the following:
      i. Create a parcel delivery order.
      ii. Change the destination of a parcel delivery order.
      iii. See the details of a parcel delivery order such as the pickup location, destination, and price.
      iv. Cancel a parcel delivery order.
      v. View all parcel delivery order the individual user has created
   c. A page/pages for a user's profile which, at minimum displays:
      i. The number of parcel delivery orders that has been delivered.
      ii. The number of parcel delivery orders that are yet to be delivered (in transit).
      iii. List of all parcel delivery orders.
   d. A page/pages where an Admin can do the following:
      i. Change the **status** of a parcel delivery order.
      ii. Change the **present location** of a parcel delivery order.

   e. The application should display a Google Map with Markers showing the **pickup location** and the **destination**.
   f. **Optional features**:
      i. The application should display a Google Map with a line connecting both Markers (**pickup location** and the **destination**).
      ii. The application should display a Google Map with computed travel distance and journey duration between the **pickup location** and the **destination**.

2. **On Pivotal Tracker create stories to build out your frontend with vanilla Javascript.**
3. **On Pivotal Tracker, create stories to capture any other tasks not captured above. A task can be feature, bug or chore for this challenge.**
4. **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. Alternatively, create a new repo in which you will develop your front end, then setup the linting library and ensure you configure the style guide properly.**
5. **Deploy your front-end to Github-Pages.**
6. **Implement your front-end functionality.**
7. **On Pivotal Tracker create stories to capture any other tasks not captured above. The tasks can be feature, bug or chore in this challenge**

   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.*

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 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

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 HTML/CSS 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 UI components on request. Applies UI/UX and coding best practices for HTML/CSS/Javascript

© Andela Confidential