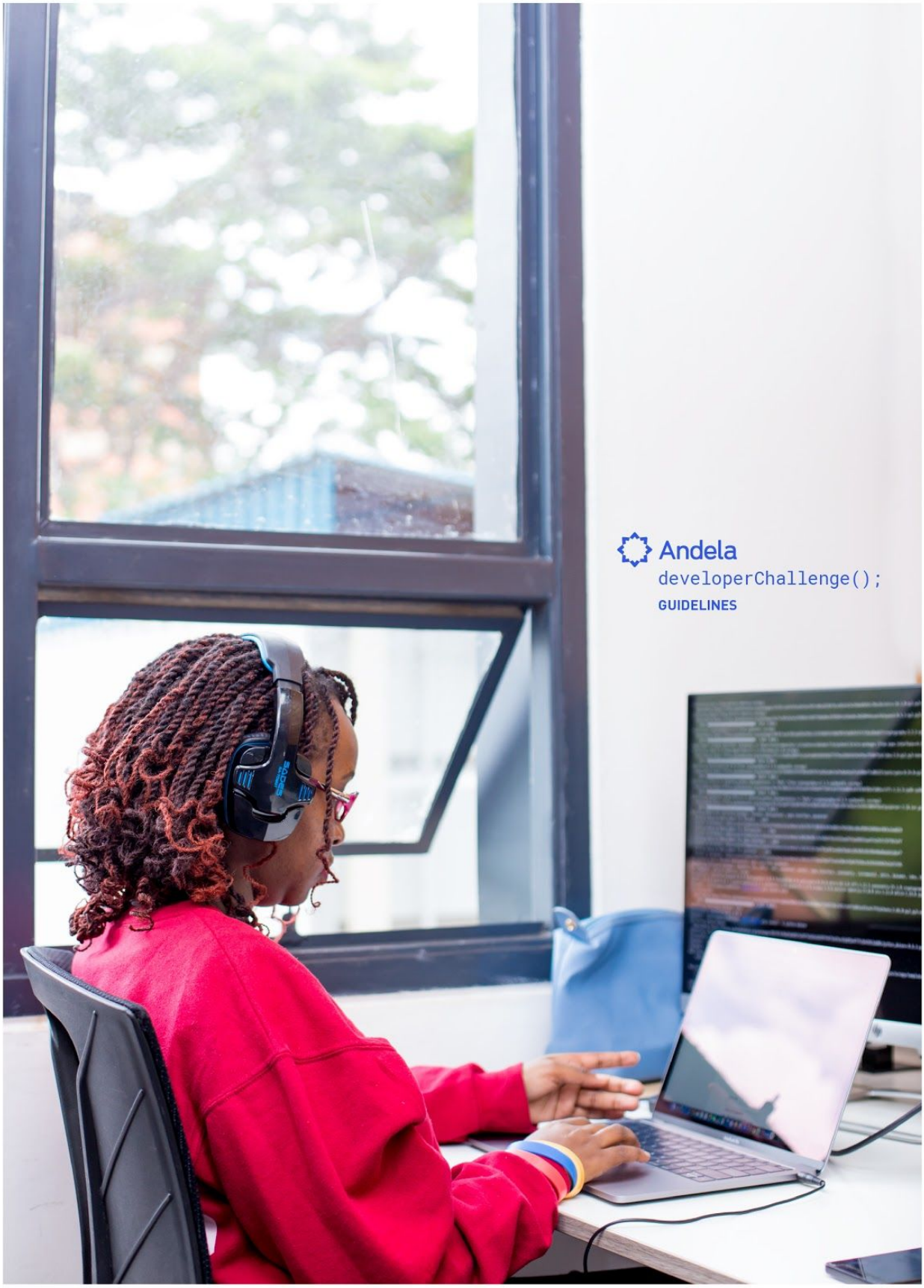
Andela



Andela

developerChallenge();

GUIDELINES

Andela

Andela Developer Challenge

Build A Product: **iReporter**

Andela

BUILD A PRODUCT: iReporter

Project Overview

Corruption is a huge bane to Africa's development. African countries must develop novel and localised solutions that will curb this menace, hence the birth of iReporter. iReporter enables any/every citizen to bring any form of corruption to the notice of appropriate authorities and the general public. Users can also report on things that needs government intervention

Project Timelines

- **Total Duration**: 4 weeks
- **Final Due Date**: Friday, 14th December, 2019

Required Features

1. Users can create an account and log in.
2. Users can create a **red-flag** record (An incident linked to corruption).
3. Users can create **intervention** record (a call for a government agency to intervene e.g repair bad road sections, collapsed bridges, flooding e.t.c).
4. Users can edit their **red-flag** or **intervention** records.
5. Users can delete their **red-flag** or **intervention** records.
6. Users can add geolocation (Lat Long Coordinates) to their **red-flag** or **intervention** records**.**
7. Users can change the geolocation (Lat Long Coordinates) attached to their **red-flag** or **intervention** records**.**
8. Admin can change the **status** of a record to either **under investigation, rejected** (in the event of a false claim) or **resolved** (in the event that the claim has been investigated and resolved)**.**

Optional Features

1. Users can add images to their **red-flag** or **intervention** records, to support their claims.
2. Users can add videos to their **red-flag** or **intervention** records, to support their claims.
3. The application should display a Google Map with Marker showing the red-flag or intervention location.
4. The user gets real-time email notification when Admin changes the **status** of their record.

5. The user gets real-time sms notification when Admin changes the **status** of their record.

NB:

1. The user can only change the **geolocation** (Lat Long Coordinates) of a **red flag/ intervention** record when the record's **status** is yet to be marked as either **under investigation, rejected,** or **resolved**.
2. The user can only edit or delete a **red-flag**/ **intervention** record when the record's **status** is yet to be marked as either **under investigation, rejected, or resolved**.
3. Only the user who created the **red-flag** or **incident** record can delete the record.

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 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

Challenge 1 - Create UI Templates

Challenge Summary

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

Timelines

- **Duration:** 1 week
- **Due Date:** Friday 23rd November, 2018

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 frameworks e.g Bootstrap, Materialize, sass/scss.*
- *Do not download or use an already built website template.*

Guidelines

1. **On Pivotal Tracker, create user stories to set up the User Interface elements:**
    a. User sign-up page.
    b. User sign-in page.
    c. A page/pages where a user can do the following:
        i. Create a **red-flag** record.
        ii. Create an **intervention** record.
        iii. Delete a **red-flag** or **intervention** record.
        iv. Add images to either **red-flag** or **intervention** record.
        v. Add videos to either **red-flag** or **intervention** record.
        vi. Add geolocation (Lat Long Coordinates) to either a **red-flag** or **intervention** record.
        vii. View all **red-flag** or **intervention** records an individual user has created.
    d. A page/pages for a user's profile which, at minimum displays:
        i. The number of **red-flag/intervention** records that has been **resolved**.
        ii. The number of **red-flag/intervention** records that are yet to be resolved (in **draft** or **under investigation** states).
        iii. The number of **red-flag/intervention** records that has been **rejected.**
        iv. List of all **red-flag/intervention** records.
    e. A page/pages where an Admin can do the following:
        i. Change the **status** of a **red-flag/intervention** records.
        ii. List of all **red-flag/intervention** records created by all users.

2. **On Pivotal Tracker, create stories to capture any other tasks not captured above. A task can be feature, bug or chore for this challenge.**
3. **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**
4. **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.*

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

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.

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:** 1 Week
- **Due Date:** Friday, 30th December, 2018

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. **On Pivotal Tracker, create user stories to setting up and testing API endpoints that do the following using data structures:**
    - Create a **red-flag** record
    - Get all **red-flag** records
    - Get a specific **red-flag** record
    - Edit a specific **red-flag** record
    - Delete a **red-flag** record
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 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. **Setup the test framework.**
7. **On Pivotal Tracker create stories to capture any other tasks not captured above. The tasks can be feature, bug or chore for this challenge.**
8. **Setup the server side of the application using the specified framework**
9. **Ensure to test all endpoints and see that they work using Postman.**
10. **Integrate TravisCI for Continuous Integration in your repository (with *ReadMe* badge).**
11. **Integrate test coverage reporting (e.g. Coveralls) with a badge in the *ReadMe*.**
12. **Obtain CI badges (e.g. from Code Climate and Coveralls) and add to *ReadMe*.**
13. **Ensure the app gets hosted on Heroku.**
14. **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 ***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
Test-driven development	Writing tests for functions or features.	
Data structures	Implement non-persistent data storage using data structures.	
Continuous Integration	Using tools that automate build and testing when the code is committed to a version control system.	
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

Andela

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.
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 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

Entity Specification

User

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

Incident

```
{
  “id” : Integer,
  “createdOn” : Date,
  “createdBy” : Integer, // represents the user who created this record
  “type” : String,       // [red-flag, intervention]
  “location” : String,   // Lat Long coordinates
  “status” : String,     // [draft, under investigation, resolved, rejected]
  “Images” : [Image, Image],
  “Videos” : [Image, Image],
  “comment” : String
   ...
}
```

Andela

API Endpoint Specification

Endpoint: GET /red-flags
Fetch all **red-flag** records.

Response spec:

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

Endpoint: GET /red-flags/<red-flag-id>
Fetch a specific **red-flag** record.

Response spec:

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

Endpoint: POST /red-flags
Create a **red-flag** record.

Response spec:

```
{
  “status” : Integer,
  “data” : [{
    “id” : Integer,    // red flag record primary key
    “message” :  “Created red-flag record”
  }]
}
```

Endpoint: PATCH /red-flags/<red-flag-id>/location
Edit the location of a specific red-flag record.

Response spec:

```
{
  “status” : Integer,
  “data” : [{
     “id” : Integer,  // red-flag record primary key
     “message” : “Updated red-flag record’s location”
  }]
}
```

Andela

Endpoint: PATCH /red-flags/<red-flag-id>/comment

Edit the comment of a specific red-flag record.

Response spec:

```
{
  “status” : Integer,
  “data” : [{
     “id” : Integer,  // red-flag record primary key
     “message” : “Updated red-flag record’s comment”
  }]
}
```

Endpoint: DELETE /red-flags/<red-flag-id>

Delete a specific red flag record.

Response spec:

```
{
  “status” : Integer,
  “data” : [{
    “id” : Integer,       // red-flag record primary key
    “message” :  “red-flag record has been deleted”
  }]
}
```