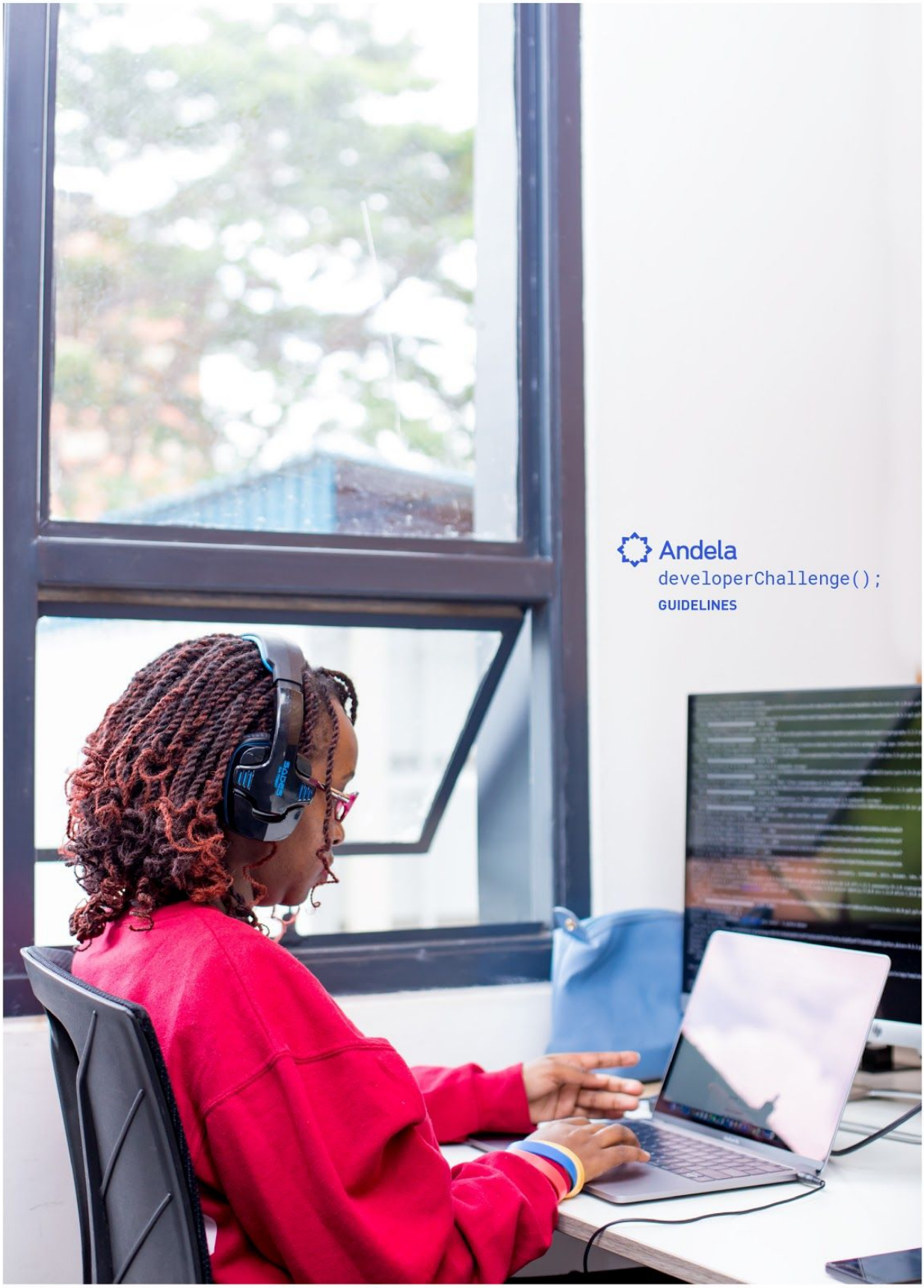
Andela



Andela

developerChallenge();

GUIDELINES

Andela

Andela Developer Challenge

Build A Product: **MyDiary**

Andela

BUILD A PRODUCT: MyDiary

Project Overview

MyDiary is an online journal where users can pen down their thoughts and feelings.

Required Features

1. Users can create an account and log in.
2. Users can view all entries to their diary.
3. Users can view the contents of a diary entry.
4. Users can add or modify an entry.

Optional Features

1. Users can set and get daily notifications that prompt them to add an entry to their diary.

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

Timelines

- **Expected Length to Complete: 1 week**
- **Due Date:** 13th of July

Challenge Summary

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

NB:

- *You are not implementing the core functionality yet, you are only building the User Interface elements, pages and views!*
- *You are to create a pull request to elicit review and feedback for the UI template when you are done working on them*
- *Do not use any css frameworks e.g bootstrap, materialize .*
- *Do not download or use an already built website template.*

Guidelines

1. **On Pivotal Tracker, create user stories to setup the User Interface elements:**
    a. User signup and signin pages.
    b. A page/pages where a user can do the following:
        i. View a list of his or her diary entries.
        ii. View the content of an entry.
        iii. Add and modify an entry.
    c. A page/pages for a user's profile which would contain information like:
        i. Reminder settings.
        ii. Number of entries since registration.

2. **On Pivotal Tracker create stories to capture any other tasks not captured above. The tasks 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.**

Andela

> ***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 learnt and be able to demonstrate the following skills.

Skill	Description	Helpful Links
Project management	Using 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 an 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

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
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.

Andela

Front-End Development	Fails to develop specified HTML/CSS web pages or uses an already built out website template, or output fails to observe valid HTML document structure	Successfully develops HTML/CSS webpages 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

Timelines

- **Expected Length to Complete: 1 week**
- **Due Date:** 20th of July

Challenge Summary

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

NB:

- *You are to create a pull request to elicit review and feedback when you are done working on this challenge.*
- *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.***

Tools

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

Guidelines

1. **On Pivotal Tracker, create user stories to setup and test API endpoints that do the following using data structures**
    - Get all diary entries.
    - Get a specific diary entry.
    - Add an entry
    - Modify an entry.
2. **On Pivotal Tracker create stories to capture any other tasks not captured above. The tasks can be feature, bug or chore for this challenge.**

3. **Setup the server side of the application using the specified framework.**
4. **Setup linting library and ensure that your work follows the specified style guide requirements.**
5. **Setup test framework.**
6. **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.**
7. **Using separate branches for each feature, create version 1 (v1) of your RESTful API to power front-end pages**
8. **At minimum, you should have the following API endpoints working:**

EndPoint	Functionality
`GET /entries`	Fetch all entries
`GET /entries/<entryId>`	Fetch a single entry
`POST /entries`	Create an entry
`PUT /entries/<entryId>`	Modify an entry

9. **Write tests for the API endpoints**
10. **Ensure to test all endpoints and see that they work using Postman.**
11. **Integrate TravisCI for Continuous Integration in your repository (with *ReadMe* badge).**
12. **Integrate test coverage reporting (e.g. Coveralls) with badge in the *ReadMe*.**
13. **Obtain CI badges (e.g. from Code Climate and Coveralls) and add to *ReadMe*.**
14. **Ensure the app gets hosted on Heroku.**

Target skills

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

Skill	Description	Helpful Links
Project management	Using 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 an sample template for creating Pivotal Tracker user stories.

Andela

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.	*<Insert resources>*
Data structures	Implement non-persistent data storage using data structures.	*<Insert resources>*
Continuous Integration	Using tools that automate build and testing when code is committed to a version control system.	*<Insert resources>*
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 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	Adheres recommended GIT workflow and uses badges.

		commit 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 optimisations.
Test-Driven development	Unable to write tests.	Writes tests that fail.	Writes tests that pass.
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 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.	
Data Structures	Fails to implement CRUD or Implements CRUD with persistence	Implements CRUD without persistence	Uses the most optimal data structure for each operation