Main Instructions

This document contains multiple assessments, complete the one that best fits your role. You would only be considered for the tasks you complete.

Submission

- It would help if you help if you added the collaborators when you're ready for review.
- It would help if you open an issue, add your name and email.
- When your work is reviewed, the issue should have a comment from a reviewer.
- You can ping any of the collaborators in your issue when you're ready for review.

DevPlacement Assessment - Web

Task: Build the dashboard with the design below

Resources

Design: XD
API: Random User Generator | Home
UX: Video

Instructions

- Use any Client Side framework of your choice.
- Write tests
- Deploy on a static hosting site.
- Make all the features shown in the video work.

Submission

- Create a private GitHub repo with your work and add @bondz, @dreplica, and @markeu as a collaborator.
- In your readme
    - Put a link to the deployed site
    - Add a badge to your CI with tests passing
    - Describe the technologies you used in the project
    - Any other information about the project you wish to include

Evaluation

- Features shown in the video work
- Commit in chunks - git history should show progress
- Subtle microinteractions included the ui - make it delightful
- Implement local search (the api does not support it)
- The UI to show countries is at your discretion.
- Pagination should work and be deterministic
- Download the current page to CSV (noinfo)
- Code is clear and easy to extend.
- Write tests for the features you implement

DevPlacement Assessment - Mobile

Resources

1. An excel file which contains the records of car owners in the United States over several years. You are to download this list and save it on your device in a folder called **owners**. Here is the link to download the file [https://drive.google.com/file/d/1giBv3pK6qbOPo0Y02H-wjT9ULPksfBCm/view].
2. A GET API which returns a list of users. Here is the link to the API endpoint [https://android-json-test-api.herokuapp.com/accounts]
3. Mockup: PDF

Instructions

- Use any of the following (Java, Kotlin, Objective C, Swift, React Native, Flutter)
- If any mobile platform you wish to use is not listed above, please email info@decagonhq
- Your app should have at least three pages.
    - One to show cars and filter them using one or more criteria
    - The second showing the list of users fetched from the API
    - The third showing additional information about the user clicked
- Save the csv file locally on your device in a folder called **owners**
  Please do not rename the file or change the format. Handle cases for when this file is missing
- Design a master-detail esque view for the users returned by api.
- Handle errors gracefully
- Write tests
- Find a good user interface online which you can use. You can search on Dribble or Pinterest for ideas.

Submission

- Create a private GitHub repo with your work and add @bondz, @darothub, and @wptechprodigy as a collaborator.
- Create a GitHub release and add the apk or ipa of your work as a binary.
- Your app name should be your full name
- In your readme
    - Add a badge to your CI with tests passing
    - Describe the technologies used in your project
    - Add any other information about your project you wish to include.

Evaluation

- The app is installable on Android (10 and above) or iOS (13 and above)
- The design is modern and UX is decent.
- Navigation is easy and seamless
- Commit in chunks - git history should show progress
- A list of cars with filters is displayed when the app is open.
- The user can filter the list with one or more criteria
- If the filter returns an empty list, the user should be informed
- The app does not crash for any user action
- Write tests for the features you implement

DevPlacement Assessment - CLI

Resources

- Data: CSV
  - This folder contains example input and example output files.
  - The input folder contains a list of *dirty* emails
  - The output folder contains examples of what the output should look like.
- MX: CloudFlare Wiki

Instructions

- Use any of the following languages (C, C++, Rust, go)
- If any mobile platform you wish to use is not listed above, please email info@decagonhq
- Your task is to design a cli program.
- Your cli program when run without any input or flags should output your name
- Your cli program should be named **your fullname** in lowercase and hyphen **(example john-doe)**
- Your program should behave as follows in the following scenarios
  - john-doe
    - No input or flags - Print your name and email
  - john-doe small-sample.csv --output small-sample.json
    - The input file should be read and it should produce a small-sample.json file. This should only validate that the emails are correct and produce an analysis of how many emails was processed. See output/small-sample.json in the Resources given.
  - john-doe small-sample.csv --output small-sample.csv --extended
    - The input file should be read and it should produce a small-sample.csv file. This should validate that the domain part of each email in the input has a valid MX record. The program should write only valid emails to the output file. See output/small-sample.csv in the Resources given.
- Your program should be able to handle large input files.
- Your program should be able to print useful help documentation.
- Write tests

Submission

- Create a private GitHub repo with your work and add @bondz, @dreplica, and @markeu as a collaborator.
- Create a GitHub release and add your work as a binary.
- Your app name should be your full name separated by hyphen
- In your readme
  - Add a badge to your CI with tests passing
  - Describe the technologies used in your project
  - Add any other information about your project you wish to include.

Evaluation

- The program behaves correctly as described in the instructions.
- Commit in chunks - git history shows progress
- Testable and extendable code
- The program doesn't crash
- The program has a good caching strategy to not overwhelm the api
- The program can handle files with 100k emails and above

DevPlacement Assessment - Backend

Resources:

Currency Conversion: API Documentation - Fixer

Instructions

- Use any of the following languages (Python, PHP, go, Node, C#, Java, Kotlin)
- If any mobile platform you wish to use is not listed above, please email info@decagonhq
- You can use any communication protocol of your choice (REST, GRPC, Graphql, etc)
- You can use one or more databases of your choice. Document the reason for your choice(s) in your architecture.
- Your task is to design a wallet system for a product used in multiple countries.
- This system would only be accessible to authenticated users.
- User types
    - Noob
        - Can only have a wallet in a single currency selected at signup (main).
        - All wallet funding in a different currency should be converted to the main currency.
        - All wallet withdrawals in a different currency should be converted to the main currency before transactions are approved.
        - All wallet funding has to be approved by an administrator.
        - Cannot change main currency.
    - Elite
        - Can have multiple wallets in different currencies with a main currency selected at signup.
        - Funding in a particular currency should update the wallet with that currency or create it.
        - Withdrawals in a currency with funds in the wallet of that currency should reduce the wallet balance for that currency.
        - Withdrawals in a currency without a wallet balance should be converted to the main currency and withdrawn.
        - Cannot change main currency
    - Admin
        - Cannot have a wallet.
        - Cannot withdraw funds from any wallet.
        - Can fund wallets for Noob or Elite users in any currency.
        - Can change the main currency of any user.
        - Approves wallet funding for Noob users.
        - Can promote or demote Noobs or Elite users
- Write concise api documentation for your endpoints
- Write tests to cover all scenarios that you implement
- Write a docker-compose file to startup your application and start your db

Submission

- Create a private GitHub repo with your work and add @bondz, @dreplica, and @markeu as a collaborator.

- In your readme
    - Add a badge to your CI with tests passing
    - Add a link to your docs
    - Describe the technologies used in your project

Evaluation

- Program can be started with instruction in the readme with docker-compose
- Program can be seeded with an admin user with the password **01234Admin**
- Program meets the specifications with tests
- Commit in chunks - git history shows progress
- Clean and extendable code
- Errors are gracefully handled and communicated to the client
- Architecture is documented
- Documentation is clear and covers the api with examples (Swagger, postman, markdown, etc)
- Users can signup or login (Social login allowed)
- Apis exist that cover the scenarios in the instruction.