Jellyvision Ruby on Rails/Web App Developer Audition

Welcome! This is an open-book audition and you may use any resource at your disposal except for one: biological entities. That is, you can't ask live people to interactively help you. This includes Skype, IRC, or the person in the next room.

Since you're performing the audition remotely, we have no idea if you're getting human help or not, but we trust that you'll follow the rules since you've gotten this far in the Jellyvision interview process.

Notes:

1. The sections may be worked on in any order you choose. There are no dependencies. However, there may be problems to solve that aren't specifically listed below that might be encountered along the way. You should solve those too, and document what you did.
2. If you manage to lock yourself out of the testing servers, it's game over (unless of course the cause wasn't your fault).
3. You may ask the proctor for clarification of an item only as a last resort. We wrote the problems the way we did for a reason.
4. You might not finish the audition, and that's ok.
5. The proctor will issue you an SSH key to access the servers listed below.
6. If you complete the audition, or decide you've had enough of us, let the proctor know.
7. Finally, we thank you for putting up with our exhaustive interview process. We know it can be grueling, but we want to make sure that you're right for Jellyvision, and Jellyvision is right for you. You'll hear from us in a few days about what the next steps will be.
8. Good luck!

Audition Stack:

1. Ruby 1.9.3
2. Ruby on Rails 3.2.x
3. SQLite
4. nginx

Optional:

1. rvm
2. guard

Gems:

You are free to include additional **gems** as you see fit, however any gems that require **external dependencies** such as sidekiq (redis) are not recommended.

Other:

These instructions are also available in markdown inside README.md.

SOLUTIONS.md should contain your answers/documentation.

The Setup:

Initial Configuration:

In order to pull down **git repo**, you will need to do some configuration on your local machine:

1. Move into the directory with the **identity_file** and copy **audition-rails.pem** (may be named something else) into **~/.ssh**

```
cp audition-rails.pem ~/.ssh
```

2. Update permissions on the **identity_file**

```
chmod 600  ~/.ssh/audition-rails.pem
```

3. Go to **~/.ssh/config** and add the following:

```
Hostname 54.214.2.238
User ec2-user
IdentityFile ~/.ssh/audition-rails.pem
```

4. Clone the git repo

```
git clone ec2-user@54.214.2.238:/opt/git/rails-web-app-developer-v2
```

Getting Started:

```
bundle install
rake db:migrate
rake seed:teams
rake seed:players
rake seed:performances
rails server
```

ssh:

```
ssh ec2-user@54.214.2.238
```

root:

No password needed.

```
sudo su
```

Final Note:

After you have completed the audition, if you decided to set up ~/.ssh/config, you should delete the file or remove the configuration settings you added related to this audition.

The Challenges:

We have a set of challenges for you. Show us what you can do and good luck, try to finish as much as you can! The challenges are broken down into the following sections:

1. **SECTION A:** Serverside (1 Problem) [REMOTE MACHINE]
2. **SECTION B:** The Basics (2 Problems)***** [LOCAL]
3. **SECTION C:** Troubleshooting (1 Problem) [LOCAL]
4. **SECTION D:** Refactoring... (1 Problem) [LOCAL]
5. **SECTION E:** Ruby (1 Problem) [LOCAL]
6. **BONUS** (optional)

***NOTE:** In *Section B: The Basics*, you need to do 2 of the provided problems. 1 problem in *Option 1 or 2* and one in *Option 3 or 4*.

You should attempt to complete each section, the required amount of problems to complete are in parenthesis.

You are free to do the sections/problems in any order you choose but be aware of the time constraints.

SECTION A is a task that must be done on a remote EC2 instance, we have provided you access to.

SECTION B – E, are to be done on your local machine. When you are done **commit** your changes and **push** them upstream. That will conclude this audition.

LAST NOTE

If you are doing the **un-timed** version of this challenge, you must do **all** of the problems.

Check with your recruiter or proctor to verify whether you are doing the **timed** or **un-timed** challenged.

1. SECTION A: Serverside (Pick 1) – [REMOTE]

Option 1: Troubleshooting: Fixing Somebody Else's Mistakes

- Server: 54.214.2.238
- User: ec2-user
- Server directory: /srv/www/audition

You will need to **ssh** into this machine, see **The Setup** for details.

Note: Some items may require a command line and/or sudo usage to correct.

There is something wrong with each of the following URLs, the webserver **nginx** is not working as expected. Fix the issues, and document what the problem was and what you did to fix it. Update **SOLUTIONS.md** with your answers.

1. http://54.214.2.238/a
2. http://54.214.2.238/b
3. http://54.214.2.238/c
4. http://54.214.2.238/d

2. SECTION B: The Basics (Pick [Option 1 or 2] AND [Option 3 or 4]) - [LOCAL]

In **Section B: The Basics**, **you need to do 2 of the provided problems:**

- 1 problem in Option 1 or 2
- 1 problem in Option 3 or 4

PICK OPTION 1 or 2

Option 1: Server Javascript Response (SJR)

Task: Let's do some AJAX with Rails. Go to http://localhost:3000/players, we have a paginated table (using kaminari). Instead of rendering a new page, make it so when clicking on a pagination button, it triggers a **remote** request via AJAX. The application should respond with javascript, the **javascript response** should replace the table with the new results.

Option 2: Building JSON

Task: Now let's **build** some **JSON**! If we look at http://localhost:3000/players.json, the JSON response could look nicer. We want the JSON response to look like the following:


```
[
  {
    "id": 1,
    "uuid": "D9AD1E6D-4253-4B88-BB78-0F43E02AF016",
    "created_at": "2014-07-11T03:22:11Z",
    "updated_at": "2014-07-11T03:22:11Z",
    "bio": {
      "first_name": "Nicolas Alexis Julio",
      "last_name": "N'Koulou N'Doubena",
      "nickname": "N. N'Koulou",
      "name": "Nicolas Alexis Julio N'Koulou N'Doubena",
      "age": 24,
      "birth_city": "Yaound\u00e9",
      "birth_country": "Cameroon",
      "birth_day": "1990-03-27T00:00:00Z",
      "nationality": "Cameroon"
    },
    "image": "http://cache.images.globalsportsmedia.com/soccer/players/150x150/39105.png",
    "stats": {
      "foot": "Right",
      "height_cm": 180,
      "position": "Defender",
      "weight_kg": 77
    },
    "team": {
      "team_id": 2,
      "team_uuid": "DF25ABB8-37EB-4C2A-8B6C-BDA53BF5A74D",
      "team_name": "Cameroon"
    }
  }
  ...
]
```


OBSERVE: The **two new fields**, **name** and **team_name**. **name** should be ***first_name and last_name.***

PICK OPTION 3 or 4

Option 3: Nested Forms and Attributes

TASK: Now let's head over to http://localhost:3000/teams/new. At this page, make it so the form will create a new **team** and a **player** (just one is fine) at the same time. The **player** should be associated to the **team**.

Option 4: Background Jobs

TASK: Let's go back to http://localhost:3000/player/1. At this page, there is a **Report** button which triggers the creation of a **report.txt** file inside **tmp**. Move this action into a **background job**. We would recommend **sucker_punch**, **Spawnling**, **DelayedJob**, or simply use **ruby threads**. We'll leave it up to you to decide. You will need to add the gem yourself if you decide to use one.

SECTION C: Troubleshooting (Pick 1) - [LOCAL]

Option 1: Where's my page dude?

TASK: http://localhost:3000/players/42/edit is broken! Figure out why and fix it. Be sure to update **SOLUTIONS.md** with your solution and why it was happening.

Option 2: Circular Dependencies

QUESTION: What is a **Circular Dependency** and when would something like this occur? Update **SOLUTIONS.md** with your explanation and a possible solution for resolving it.

Option 3: Inheritance

TASK: Something seems to be wrong with **Sponsor::Gold**, resolve the issues and update **SOLUTIONS.md** with the reasons for why the **Sponsor::Gold** wasn't working and your solution. There are a total of 3 issues.

SECTION D: Refactoring... or is it more like rewriting? (Pick 1) - [LOCAL]

Option 1: Queries on Queries

TASK: The **queries** being executed in **Team.get_players** is incredibly inefficient. Refactor the entire method being run. **Team.get_players(team_ids),** should take an array of **team_ids**, and it should return an **array** of **hashes** formatted as follows:

```
[
  { id: 1, name: 'Darijo Srna', position: 'Defender', team: 'Croatia' },
  ...
]
```

OBSERVE: **name** should be ***first_name and last_name.***

Option 2: Abstraction, DRY, and Best Practices

TASK: FAT MODELS! SKINNY CONTROLLERS! Take a look inside **StatisticsController**, **statistics/views/index.html.erb**, **Player** and **GamePerformance** and clean up the code!

NOTE: For simplicity, the **@player** is purposefully going to one player, you will not need to modify this.

SECTION E: Ruby (Pick 1) - [LOCAL]

Option 1: Webscraping and CSV Parsing

TASK: We need to fetch some data, but unfortunately there is no JSON api. We'll leave it to you how you want to parse the **HTML**, but we recommend using **nokogiri** or **mechanize**. Fetch the page http://localhost:3000/data/stats.html and convert all the data into a CSV file. Take the CSV file and add it to the database, this data corresponds to the **GamePerformance** model.

To recap:

1. Make a **http GET request** to http://localhost:3000/data/stats.html
2. Parse the **HTML** from http://localhost:3000/data/stats.html
3. Convert the data in the table into **CSV,** store it inside the **rails root folder,** name it **table.csv**.
4. Parse the **CSV** file and save it to the database via the **GamePerformance** model
5. These **methods** should exist in **GamePerformance** and we should be able to call **scrape_data** that triggers all of those actions.

BONUS (optional)

Bonus Question:

AX QGM'JW JWSVAFY LZAK LWPL, LZW SFKOWJ LG AFUDMVW AF QGMJ VGUMEWFLSLAGF AK "USWKSJ".

Place your answer in **SOLUTIONS.md**.