Introduction

Build an externally usable API for a basic to-do list application. This API will allow users to modify user accounts and to-do items from the command line.

Use Case

To-do lists should be simple, while remaining flexible to use. It's one thing to have a physically limited stack of papers in your pocket. It's quite another to have a list that works easily on your Android, and your spouse's iPad, and your Windows computer (and any command line, worldwide).

Apps like **Todo.txt** go a long way towards solving this problem by creating a simple API that any programmer can easily navigate and extend. Like Todo.txt, this app will be easy to control from the command line.

Your API should allow you to change the same data from the command line or the browser. This API can support other platforms and allow programmers to build on your tool in new and exciting ways.

User Stories

User Story	Difficulty Rating
As the Open To-do API, I want to **return** JSON representations of users, lists, and items	2

As a user, I want to **authenticate** myself from the command line, using a username and password	2
As a user, I want to **create** new users, lists, and items from the command line	2
As a user, I want to **remove** users and lists from the command line	1
As a user, I want to **update** list and item attributes from the command line	3

Before you begin working on user stories, complete this project's **Getting Started guide**. Later user stories often rely on the completion of the former, therefore, work on them in the order prescribed.

How would you rate this checkpoint and assignment?

Rails Serializers

> As the Open To-do API, I want to **return** JSON representations of users, lists, and items
> **Difficulty Rating**: 2

Generate Models

Open To-do API requires three models: a `User`, a `List` and an `Item`. A user model represents a user, an item is a single to-do item, and a list belongs to a user, has several items, and a `private` attribute. Generate these three models with basic attributes.

Generate Serializers

The Open To-do API must return formatted responses that users can read, and machines can generate and parse. The standard for most modern web APIs is **JSON**, which stands for JavaScript Object Notation. JSON is a lightweight data-interchange format.

Converting a Rails object into a JSON representation is called serializing. Open To-do API will need to serialize users, lists, and items. Read our guide to **Rails Serializers** to turn your Rails objects into JSON.

Generate `UserSerializer`, `ListSerializer`, and `ItemSerializer` using the Rails Serializers guide.

Test your code

- From the Rails console, insert at least one user, list, and item.
- From the Rails console, confirm that `puts JSON.pretty_generate(UserSerializer.new(User.first).as_json)` outputs the JSON representation of a `User`.
- From the Rails console, confirm that

`puts JSON.pretty_generate(ListSerializer.new(List.first).as_json)` outputs the JSON representation of a `List`.

- From the Rails console, confirm that `puts JSON.pretty_generate(ItemSerializer.new(Item.first).as_json)` outputs the JSON representation of an `Item`.
- Validate the JSON output for each Serializer using **JSONLint**.

How would you rate this checkpoint and assignment?

User Authentication

> As a user, I want to **authenticate** myself from the command line, using a username and password
> **Difficulty Rating**: 2

Generate a Base Controller

Create an `ApiController` from which the user, list, and items API controllers will inherent:

app/controllers/api_controller.rb

```
+ class ApiController < ApplicationController
  # #1
+   skip_before_action :verify_authenticity_token
+
+ end
```

#1: Refer to the **CSRF resource** to see why you need to skip the `verify_authentication_token`.

Create a private method named `authenticated?`. Other API controllers will use `authenticated?` to ensure users are authorized:

app/controllers/api_controller.rb

```
  class ApiController < ApplicationController
    skip_before_action :verify_authenticity_token
+   private
+   def authenticated?
  # #2
+     authenticate_or_request_with_http_basic {|username, password| User.where( usernar
+   end
  end
```

#2: `authenticate_or_request_with_http_basic` implements basic HTTP authentication, which ensures HTTP requests are accompanied by a valid username and password.

> Please note that using authenticate_or_request_with_http_basic would require your password to be stored as plaintext. In a production environment, you would want to implement **hashing** to "hash" your credentials. Alternatively, we could use the **Devise** `valid_password?` method if you are using Devise for authentication.

To test `authenticated?`, you will need an API route that requires authentication.

Edit `routes.rb` to provide API routes:

app/config/routes.rb

```
  # #3
+   namespace :api, defaults: { format: :json } do
+     resources :users
+   end
```

#3: `namespace` separates the API routes from the rest of the application routes. `defaults: { format: :json}` tells the route to support requests in JSON form.

Generate a Users Controller

Create `UsersController` to match the API routes. Make a new directory in `app/controllers` named `api`. Create the new controller in that directory:

app/controllers/api/users_controller.rb

```
+ class Api::UsersController < ApiController
  # #4
+   before_action :authenticated?
+
+   def index
+   end

+ end
```

#4: The `before` filter calls `authenticated?` before the request is processed.

Write `index` to return a `UserSerializer`-generated JSON representation of all users. The final line of the `index` method will look like:

app/controllers/api/users_controller.rb

```
    def index
      ...
+     render json: users, each_serializer: UserSerializer
    end
```

Test Your Code

- Create users via the Rails console.
- From the command line, retrieve all the users via a `curl` request. Replace `username` and `password` with a valid username and password:

  Terminal

  ```
  $ curl -u username:password http://localhost:3000/api/users/
  ```

- Try to retrieve all users using an invalid username and password combination, verify the request fails.

Create From the Command Line

> As a user, I want to **create** new users, lists, and items from the command line
> **Difficulty Rating**: 2

Edit Routes

Edit `routes.rb` to provide the API routes for `List`s and `Item`s:

app/config/routes.rb

```
    namespace :api, defaults: { format: :json } do
-     resources :users
+     resources :users do
+       resources :lists
+     end

+     resources :lists, only: [] do
+       resources :items, only: [:create]
+     end

+     resources :items, only: [:destroy]
    end
```

Create List and Item Controllers

Create `ListsController` and `ItemsController` to match the API routes:

app/controllers/api/lists_controller.rb

```
+ class Api::ListsController < ApiController
+   before_action :authenticated?
+
+   def create
+   end
+
+ end
```

app/controllers/api/items_controller.rb

```
+ class Api::ItemsController < ApiController
+   before_action :authenticated?
+
+   def create
+   end

+ end
```

In `UsersController`, add a `create` method and a private `users_params` method. `User` only requires username and password parameters:

app/controllers/api/users_controller.rb

```
  class Api::UsersController < ApiController
    ...
+   def create
+   end

+   private
+   def user_params
+     params.require(:user).permit(:username, :password)
+   end
  end
```

Use `user_params` in `create` to create and save a new `User`:

app/controllers/api/users_controller.rb

```
  class Api::UsersController < ApiController
  ...
    def create
+     user = User.new(user_params)
+     if user.save
  # # 5
+       render json: user
+     else
  # # 6
+       render json: { errors: user.errors.full_messages }, status: :unprocessable_ent
+     end
    end
```

5: When you use `render :json`, Rails searches for a serializer for the object and use it if it is available. In this case, Rails will look for a serializer named `UserSerializer` and use it to serialize `user`.

6: If saving the user fails (due to a missing user name or password), return the error messages and a **422 status code** indicating that the data sent was un-processable.

Test From Command Line

Test `create` from the command line:

Terminal

```
$ curl -u username:password -d "user[username]=Sterling" -d "user[password]=Archer" h
```

Once user creation is working, implement list creation. To test list creation, use `curl` from the command line to create a new list for the first user:

Terminal

```
$ curl -u username:password -d "list[name]=Things to do today" -d "list[permissions]=
```

Once list creation is working, implement item creation. To test item creation, use curl from the command line to create a new item for the first list:

Terminal

```
$ curl -u username:password -d "item[description]=Dance if you want to" http://localh
```

Test Your Code

- Modify the curl request for creating users to send a request without a **password**. Confirm an error message is returned, and a user is not created.
- Modify the curl request for creating users to send a request without a **username**. Confirm an error message is returned, and a user is not created.
- Modify the curl request for creating lists to send a request without a **name**. Confirm an error message is returned, and a list is not created.
- Modify the curl request for creating items to send a request without a **description**. Confirm an error message is returned, and an item is not created.

How would you rate this checkpoint and assignment?

Destroy Lists

> As a user, I want to **remove** users and lists from the command line
> **Difficulty Rating**: 1

Modify the Users Controller

Add a `destroy` method to `UsersController`:

app/controllers/api/users_controller.rb

```
  class Api::UsersController < ApiController
    ...
+   def destroy
+     begin
+       user = User.find(params[:id])
+       user.destroy
  # #1
+       render json: {}, status: :no_content
+     rescue ActiveRecord::RecordNotFound
+       render :json => {}, :status => :not_found
+     end
+   end
    ...
  end
```

1: Return HTTP 204 No Content to indicate the server successfully processed the request but is not returning any content.

Implement list deletion as well.

Test Your Code

- Test `User` deletion from the command line:

Terminal

```
$ curl -u username:password -X DELETE http://localhost:3000/api/users/1/
```

- Test `List` deletion from the command line:

Terminal

```
$ curl -u username:password -X DELETE http://localhost:3000/api/users/1/lists/1
```

How would you rate this checkpoint and assignment? ☺ ☹

Update Lists and Items

> As a user, I want to **update** list and item attributes from the command line
> **Difficulty Rating**: 3

Controllers

Allow users to change a list's `private` attribute from the command line. Add an `update` method to `ListsController`:

app/controllers/api/lists_controller.rb

```
  class Api::ListsController < ApiController
  ...
+ def update
+   list = List.find(params[:id])
+   if list.update(list_params)
+     render json: list
+   else
+     render json: { errors: list.errors.full_messages }, status: :unprocessable_entity
+   end
+ end
```

Add the ability to update items and mark them as complete to `ItemsController`.

Test Your Code

- Test list permission updates from the command line:

Terminal

```
$ curl -X PUT -u username:password -d "list[private]=true" http://localhost:3000
```

- Test item completion from the command line:

Terminal

```
$ curl -X PUT -u username:password -d "item[completed]=true" http://localhost:30
```

How would you rate this checkpoint and assignment?

Update Lists and Items

Assignment | Submission