



[Public] - Ruby - Take Home Test

Description

Your task is to create a timesheet entry application. Keep in mind that you will be required to extend it in the next process of the interview.

A timesheet consists of a **Date**, a **Start Time** and a **Finish Time**. Your application should display existing timesheets, calculate their dollar value, and allow users to create new timesheets.

It should consist of two screens - mockup shown below.

Mockup of Screen 1: Create Timesheet Entry. The screen has a title 'Create Timesheet Entry' at the top. Below the title are three input fields: 'Date', 'Start Time', and 'Finish Time'. At the bottom of the form is a 'Create' button.

Mockup of Screen 2: Timesheet Entries. The screen has a title 'Create Timesheet Entry' at the top. Below the title is a section titled 'Timesheet Entries'. Under this section, there are two entries: '07/01/2019: 10:00 - 17:00 \$154' and '08/01/2019: 14:00 - 22:00 \$230'.

Screen 1: Create Timesheet Entry

Inputs

- Date of entry

- Start time
- Finish time

Create Button

- When clicked it should attempt to create a new entry
- If successful it should redirect to the **Timesheet Entries** screen, and the newly created entry should be visible
- If it does not succeed it should display validation errors clearly on the screen and allow the user to fix them and try again

Validation

When creating a timesheet fails the errors should clearly be displayed

- You can't have overlapping timesheet entries.
- Date of entry
 - Required
 - Can't be in the future
- Start time
 - Required
- Finish Time
 - Required
 - Can't be before start time

Screen 2: Timesheet Entries

Create Timesheet Entry Button

- When clicked it should take the user to the **Create Timesheet Entry** screen

Timesheet Entries Table

- Display a list of timesheet entries entered
- In the format of `"#{date of entry}: #{start time} - #{finish time} $#{calculated amount}"`
- The dollar value should be calculated as described in the Calculation Section

Calculations

- Monday, Wednesday, Friday
 - 7am - 7pm: \$22/hour
 - Outside: \$34/hour
- Tuesday, Thursday
 - 5am - 5pm: \$25/hour
 - Outside: \$35/hour
- Weekend
 - Always \$47/hour

Calculation Examples

- 15/04/2019 10:00 - 17:00 \$154
 - Monday rate: ($\$22 * 7\text{hrs}$)
- 16/04/2019 12:00 - 20:15 \$238.75
 - Tuesday rate: ($\$25 * 5\text{hrs}$) + ($\$35 * 3.25\text{hrs}$)
- 17/04/2019 04:00 - 21:30 \$451
 - Wednesday rate: ($\$34 * 3\text{hrs}$) + ($\$22 * 12\text{hrs}$) + ($\$34 * 2.5\text{hrs}$)
- 20/04/2019 15:30 - 20:00 \$211.5
 - Weekend rate ($\$47 * 4.5\text{hrs}$)
- 17/04/2019 02:00 - 6:00 \$136
 - Wednesday rate: ($\$34 * 4\text{hrs}$)

Constraints

- You can use any library or framework (Rails, Sinatra, Grape, etc...)
- Must be written in ruby

Deliverables

- Please provide your source code, and any test code/data you use in developing your solution using `git bundle` instruction as follows:
 - In your project dir do `git bundle create ../<yourname>.bundle master`
 - Please ensure that you work on `master` branch, and replace `yourname` with your actual name, snake cased, such as `john_citizen`
 - That will create `<yourname>.bundle` file in one directory up, please submit this file
- Please engineer your solution to a standard you consider suitable for production. You will be asked to extend parts of your solution.
- **At the very minimum, ensure your tests are passing.** We are strict about this, and this is one of many criteria we considered as production quality code.
- Consider updating the Readme, use this so the reviewer can understand your code better (At least having an instruction to install your application).