CROSS | OVER

Congratulations on making it to this stage of the evaluation process! By making it to this stage, you have already proven that you are among the top candidates for this position. The technical trial will give you an opportunity to show hiring managers more than just your resume. Through this assignment, you have the potential to differentiate yourself and stand out among your peers. After you complete and deliver the objective, you will have a brief interview with one of our team members to discuss your project. Upon a successful interview, you will have completed the evaluation process and will be ready to be hired.

The project is designed to be simple and reasonable in size to enable you to demonstrate your enterprise class skills. Though this is a fictitious example, this scenario is very similar to what you may encounter on the job.

Please read through to the end before starting to work on it.

Instructions

- Try to complete as much as possible within the given time frame. If you need more time, please ask for an extension. You must complete full-functionality of the application with industry-level coding style /commenting. Unfinished assignments will not be considered.
- Please note that you are expected to work on the assignment independently. Discussing assignment details with colleagues or any indication of outside help will be considered cheating.
- Please do not expect too much hand-holding as this is an evaluation assignment.
- Read the complete assignment before you start. Understand clearly what is required so that your work will be appropriate and easier.

Overall Objective

Create the architecture and design of a Customer Support ticketing system. Implement the system's services and applications.

Prerequisites

The following prerequisites must be respected.

1. Use the Ruby on Rails technology stack for development along with any compatible open technologies.
2. Use MySQL database, and development environment tools according to the technology stack.
3. Do not use any proprietary technologies or tools that are not available for evaluation.

Functional Requirements

The system allows customers to be able to place support request and the support agents to process the requests. The system implements the following specifications.

1. For customers
    1. A web portal to create support requests and to view status of previous requests.
2. For support agents
    1. A web portal to find and process support requests.
    2. One report, on requests data, with all tickets closed in the last one month; Should be PDF exportable.

Other functional requirements

1. Users should be able to authenticate normally
2. An admin user should be able to manage other users and any other system objects.

Assume any functional details required to achieve the above requirements based on logic and your experience.

Other Technical and Non-functional Requirements

The following list of technical specifications must be adhered to

1. Use MySQL server, Ruby on Rails on back end, Bootstrap and a JS framework on front end. Use latest versions if possible.
2. On back-end : REST API, db migrations, ActiveRecord, rspec tests
3. On front-end : SPA pattern web-app, with responsive design, using a JS framework from either Angular/Ember/React/Knockout/Backbone, testing with either Qunit/Jasmine/Karma.

4. Enough coverage of proofing automated tests on both back and front end (coverage > 30%).
5. Secure the applications and services.

What we will evaluate

1. Efficacy of your submission: fundamentally how well your solution is able to achieve the assignment objective and solve the stated problem.
2. Code quality
    1. Code modularity
    2. Application organization across files and within each file - please ensure you follow the framework standards
    3. Code documentation - balancing between self documenting code and comments
    4. Unit and integration testing
    5. Exception handling where available and expected in the frameworks you're using
    6. For any technology used, the correct usage of that technology based on consecrated best practices
3. Design
    1. Clarity and completeness of the readme and design documents
    2. Fitness of solution to problem
    3. Efficiency of communication flows between front-end and back-end, if applicable
4. Functional completeness
5. Scoring ratio matrix (out of 10), all of these are individually mandatory so don't skip any:
    1. Design quality = 2
    2. Code quality = 2
    3. Docs and demo quality = 2
    4. Specifications compliance = 4

What to deliver

Demonstration Video

Record the video demonstration of your work using a screen-cast tool like Screencast-O-Matic (or any other tool you prefer) commenting on the execution of all components. Save the video to your local machine and include it with the delivery package.

Database scripts

Create manual steps and SQL script files to create the database (if required), its schema, stored procedures, or any seed data you have used for testing.

Readme Document

Create a text file with the following information

1. Instructions to install and configure prerequisites or dependencies, if any
2. Instructions to create and initialize the database (if required)
3. Assumptions you have made - it is good to explain your thought process and the assumptions you have made
4. Requirements that you have not covered in your submission, if any
5. Instructions to configure and prepare the source code to build and run properly
6. Issues you have faced while completing the assignment, if any
7. Constructive feedback for improving the assignment

Design Document

Create a design document containing the following

1. High level requirement analysis
2. High level presentation of the data model
3. Architecture diagrams describing the composition and working of the system, explaining the component interaction and process, control and data flows.
4. Explain the breakdown of the system into components with technical implementation details of each component along with the design patterns involved and with reasons that justify your choices.
5. Use both visual elements (diagrams) and text descriptions to maximize the amount of information conveyed while keeping the document as compact as possible

Source Code

You must deliver all the implemented source code including any dependencies. For the dependencies that could not be included due to size, the readme file must have proper instructions on how to download and install them.

What to submit

Please read this section carefully.

Failing to follow these directions will disqualify you from consideration.

Create and submit an archive named <*your_name*>*_SA_RubyRails_Tickets.zip* containing the following: (For example JohnDoug*_SA_RubyRails_Tickets*.zip)

- <your_name>_SA_*RubyRails_Tickets*.zip
- <your_name>_SA_*RubyRails_Tickets*.zip\Readme.txt
- <your_name>_SA_*RubyRails_Tickets*.zip\Design.doc *(or pdf, etc.)*
- <your_name>_SA_*RubyRails_Tickets*.zip\Demo\ *(this folder contains the screen-cast video recording)*
- <your_name>_SA_*RubyRails_Tickets*.zip\SQL\ *(this folder contains the SQL scripts used for setting up the database)*
- <your_name>_SA_*RubyRails_Tickets*.zip\Code\ *(this folder contains the complete source code)*

Check that the size of the archive is less than 100MB. If not, reduce the size of the demo video by removing similar frames and remove the 3rd party binary dependencies.

The following are mandatory, failing which your delivery will not be evaluated.

1. Readme document and Video demo
2. Functional automatic unit / integration tests
3. The projects should build without errors