Thank you for taking the time to do the **Carna** take-home project. It consists of 3 parts:

- Carna project
- Documentation/ReadMe
- A few technical questions

Take a moment to think—and remember, **there's no one single correct answer or approach.**

Submission Guidelines

Please review the options below to find out which suits you the best to submit the project.

⚠️ You should send the finalised project via Email to **takehome@carna.ai** address.

Option 1: As a .zip file (to avoid bounced emails, we would like you to submit your results by uploading the relevant ZIP file to a shared Google Drive folder.)

The zip file should be named {yourname}.zip and should contain the full-stack-project folder with your submission.

Additionally, the zip file should contain the Documentation/ReadMe file and Technical-Q file with answers to a few technical questions. (These can be Google Doc, MS Word, .md file or whichever you prefer)

Option 2: Host it somewhere and share the link (If you don't want to share the code with us, you can simply host the project somewhere and share the link)

Additionally, you should share the Documentation/ReadMe file and Technical-Q file with answers to a few technical questions. (These can be Google Doc, MS Word, .md file or whichever you prefer)

Carna Project

We are NOT expecting any UI for this task. Our App screens are already ready and finalised. We will share some similar screens(not Carna Screens) that are specifically designed for the take-home project with you to give an idea and boost to start if you want to use them.

Task Requirements:

Feel free to spend as much or as little time on the exercise as you like as long as the following requirements have been met. **We are not setting any boundaries in this task so that you can come up with your own ideas/choices.**

- Create a Web-based Admin Panel for either Mobile or Web App. (**Mobile App preferred.**)
  Sample features that you can show on the Admin Panel can be as follows: (only to give you some idea)
  - **Content management** (i.e. so-called CRUD: create, read, update, and delete)
    For example, Admin should be able to add New Course Content to Course Screens (some sample Course Screens are provided down below)
    Tip: Choose a framework that takes care of that for you, and don't lose time.
  - User management (e.g. adding, deleting, creating groups, enabling/disabling user privileges etc.)
  - Export/import of data (i.e. basic integration with other systems)

- Create 4-5 App Screens (**or more if you have time & Mobile preferred**) that communicate with the Backend, Database and Admin Panel.
  We provide some sample screens below, but you can feel free to develop your own ideas.

  👇**Click to see the sample screens.**👇
  SAMPLE SCREENS

Platform Choice:

You can create the application as either a command-line application, web application or mobile application on any of the following platforms: **(mobile application preferred)**

- .NET, Python or JavaScript/TypeScript (can use Angular/React/Vue.js) for web applications **(React preferred)**
- .NET or Python for command-line applications
- React Native, Flutter, Swift, Objective-C, Java, Kotlin for **Mobile Applications (React Native preferred)**

(Backend: **Node.js** or **Django** preferred.)
Database: **PostgreSQL** preferred, but you can use any SQL Database or even NoSQL.

Think about the type of work you would like to do at **Carna** and choose an appropriate application type and platform.

Other Notes/Extras:

- Your code should compile and run in one step.
- Feel free to use whatever frameworks/libraries/packages you like.
- You must include tests.
- Please avoid including artifacts from your local build (such as NuGet packages or the bin folder(s)) in your final ZIP file.
- If you still have questions that what to include in your admin panel **as extra,** here are some more ideas that you can pick from:
  - Admin will be able to login into the system using API. You can hardcode the admin user name and password. Mention it in the readme. Use JWT for Authentication.
  - Admin will be able to create Users. Modify their details. View all the users or a specific user in the system and delete a specific user.

[Nice to have] Filters for List of Users (which school, district, city, country etc.)
[Nice to have] End to End tests.
[Nice to have] Auto deployment scripts
[Nice to have] Comments in code
[Nice to have] Load test script for 100 concurrent requests.

💡 Tips for Documentation/ReadMe

Documenting the code is a MUST in any software development process. Here are some tips for you:

- **Notes on starting the app** or setting it up, for the reviewer.
- **Summary of what was built and what was (deliberately) not.** It saves a lot of nitpicking when, e.g. the reviewer knows that of the bonus tasks, pagination was implemented, but lazy loading was not. And it helps call attention to all the extra work you've done. Did you put in additional creativity to handle one of the edge cases? Write it down; else, the reviewer might completely skip it.
- **Context on decisions and framework choices.** It is good to give context if you've tried out a new framework or why you've chosen a specific architecture pattern. This is especially true for more experienced candidates. Don't underestimate the impact: reviewers will take the context into account. They are less likely to be super strict if they understand you experimented with something new.
- **Areas for improvements.** You probably ran out of time and could have done a lot more. So list these things! It goes a long way when reviewers see that you knew what else was missing. It's also good to mention what the things are that you would have worked on if this was a project that went into production.

Technical Questions:

If you already answered these questions before in your Documentation, in that case, you can just copy and paste the part from your documentation to the related question.

- What libraries did you add to the frontend? What are they used for?
- What's the command to start the application locally?
- How long did you spend on the coding project? What would you add to your solution if you had more time? If you didn't spend much time on the coding project, then use this as an opportunity to explain what you would add.
- What was the most useful feature that was added to the latest version of your chosen language? Please include a snippet of code that shows how you've used it.
- How would you track down a performance issue in production? Have you ever had to do this?

Thanks for your time. We look forward to hearing from you!

Note: “The exercise is graded against a rigorous set of over 20 predetermined criteria. We’re looking for code that is clean, readable, performant, and maintainable. We put significant effort into developing these criteria to ensure that, regardless of who grades the exercise, the score is an accurate reflection of the quality of the work.”