Quick Quotes Calculator

The Quick Quotes Calculator (QQC) is a mobile app, which aims to streamline the process of quoting jobs. This will primarily be targeted at tradespeople and sole traders, who regularly need to quote complex jobs. The app would give the user the ability to quickly tally up the quantity and cost price for each line item of any given task, with users able to customize job names and types making it a more streamlined and accurate process. This information is used to instantly generate an itemized quote that is individualized and able to be exported to other apps such as email for ease of customer communication.

According to Hipages, 2023, “trade workers spend an average of seven hours a week preparing quotes”. This equates to nearly 20% of the entire working week. Much of this time is spent summing up quantities, and total values for each job item. If the tradesperson were to make any errors in calculations, many hours may be spent trying to find the mistake. This figure will likely grow as jobs get more complex, and Australia’s math skills decline (Karp, 2019). By instantly generating quotes, QQC greatly mitigates any risk of mis-calculation, saving time and money for the business.

The app itself provides a database of all previous quotes generated by the user. From this UI, the user can easily view a list of each previous quote, complete with a basic summary of the job name, number of items on the quote, and the total quote price. This basic information about each quote helps the user easily find previous jobs, and quickly view the total quote value. This is useful for quickly finding and comparing previously entered quotes. The user may also wish to export a quote to a PDF, for seamless generation of customer quotes and receipts. This feature will likely be the primary source of monetisation within the app, as it is not required for the app to work, but may be highly desirable for some businesses who wish to further streamline their workflow. From here the user may choose to view an existing quote by selecting the quote they wish to view, or generate a new quote by selecting the *New Quote* option – both routes will navigate the user to the *Quote View* window. In *Quote View*, the UI displays a list of job items, which each include an *Item name, price per unit,* and *total price.* At the top of the UI, the job/quote name is displayed, along with a *Total quote value* dollar figure which is updated every time any changes are made within the quote. For a new quote, these fields are filled with default values, and only one line item is added to the list. The default values act to give the user some pointers on where their desired job information should go (i.e. the default value for the overall job title would be “Job Title”). If the user has navigated here by selecting a previous quote, an appropriate number of line items are automatically added to the list, and all fields are populated as per the selected quote. The key to this apps power is frontend component of this widow. All line items are stored in a scrollable list, for easy viewing of limitless line items. Each line item has editable fields, and also includes arrows to the left and right of the *quantity* counter. This means that the user may simply press a button to add or subtract from the quantity of a line item, rather than having to select and edit the text inside the *quantity* field. The difference between a button interface and text input field seems trivial at first glance, but actually greatly streamlines workflow, and is one of the key features of our app. Toward the bottom of this window will be options to delete the quote from the database, discard any changes, or save as a new quote. All of these features would be presented in a clean, easy-to-read format, with high contrast elements. This ensures maximum user accessibility and readability, making it and easy tool for anyone to use.

In a case where an electrician was required to generate a quote for installing lights and fans at a commercial premises, their quote may include quantities that run into the hundreds. Were that electrician to accidentally make an input error after counting hundreds of fans or lights, they would likely need to start all over again, potentially costing hours. This error may also easily go unnoticed, which could result in the final dollar figure of their quote being incorrect. This may cost the company money, or hurt their reputation and rapport with the customer. Aside from the factors of error elimination, this design also saves time whenever the *quantity* figure is added to or subtracted from, by providing a button-driven interface, rather than classic text input fields. Here at DualCab, we have put this test by timing how long it takes us to enter text into an input field on a mobile device. By creating and timing events within a basic text input field with unity, we found that it takes approximately 1.6 seconds to enter a single-character number into a text field. This is approximately 1.5 seconds longer than it takes to press a button. By this analysis, it is clear that our approach to numerical input has reduced this portion of the process by a factor of roughly 16 times.

For the development of this project, we will be using the Unity game engine. This platform is highly versatile, and can directly build applications for “+25 leading platforms” (Unity, 2023), including our target platforms (IOS and Android), with very little variation in the build process. This is method is far superior to using platforms such as Android Studio and XCode, because it means that we only have to create the application once. Once the app is developed, it can be deployed to both platforms by simply changing the Build settings. This is particularly important when working on a team with varying skillsets, because the learning curve for each person to adopt two new platforms is too great for this task. Although engines such as Unreal and Godot have similar perks, Unity has been selected for it’s “low entry threshold” (Unity, 2023), and its $0 price tag. Unity also has a feature called PlayerPrefs, which allows data to easily be stored. This eliminates any variations in file management between IOS and android, as well as PC which will be used regularly for testing development versions of this app. Any data stored in PlayerPrefs is also persistent across updates, meaning the user can confidently update their app without fear of losing records. All of these features are facilitated using .NET infrastructure (C#), which is regularly updated and document by Microsoft, making it a reliable and easy to learn. As C# is a low level language with very flexible, simple abstraction, it is perfect for effectively handling complex data classes, which is key to smooth, correct, and expandable functionality of this app. While Unity is packed with many great beginner-friendly features, “You won’t find any structured UI in this engine like in others” (Neurosys, n.d.). This means that a large portion of development will go toward developing and refining quality UI features, using the limited UI Toolkit that Unity provides. As outlined in Unity’s platform documentation (Unity, n.d.), “in the current release, UI Toolkit does not have some features” which were included in superseded Toolkits. From our experience here at DualCab, the Unity UI building features are not only lacking in some ways, but are far inferior to toolkits provided in platforms such as Xcode. This will be the greatest challenge in using the Unity platform. The only part of this project that will not be completed in Unity will be the graphics, as it has no image or vector manipulation features. All image processing will be handled using Gimp or Photoshop (for bitmap), and Inkscape or Illustrator (for vector). While Gimp and Inkscape are great free open-source options, some team members may choose to use Photoshop or Illustrator. This choice come down to nothing more than the personal preference of each team member, as both alternatives have equivalent functionality. Finally, a GitHub repository will be used to contribute and track project changes amongst all members of the group. This repository should be set to private, as only the source files will be available here, and we do not wish for anyone else to be able to access these.

Throughout the development process of this app, many skills will be required, including frontend development, generation and manipulation of 2D graphics, as well as backend programming. The frontend portion of this task will require the team to create a neat, smoothly functional, easy to read interface. Due to Unity’s limited UI toolkit, great care will need to be taken to ensure the interface functions, looks, and scales exactly as intended. This will involve researching how other mobile apps are laid out, and emulating some of those design constraints. All graphics generated will need to be fit for purpose, meaning image manipulation skills will be critical to ensuring the UI looks perfect. All logos, background panels, button icons etc will need to be designed with performance, scalability, image quality and compatibility in mind. To achieve this, a wide array of digital image manipulation skills are required. In contrast to the frontend development, the backend will require a strong knowledge of the OOP paradigm, and some understanding of .NET/C#. These skills would be greatly complimented by an aptitude for logic and math, for cleanly, accurately, and efficiently handling complex data. These backend skills will also determine how fast or slow the final product will be, as poor programming practices are particularly harmful to the performance of mobile devices. Because the required skills are so varied between each portion of this project, our iterpoersonal communication skills will be essential to coordinating and integrating each element of the task. According to Bootcamp, n.d., the top skills required for working on a team are “Listen before speaking”, “Speaking skills”, “Speaking a common language”, and “Confidence in body language”. A mastery of these skills will greatly optimize workflow, and help to produce a high quality product before the given deadline.

# References

Bootcamp, n.d. *8 Skills you must have as a Front-End Developer in 2022.* [Online]   
Available at: https://bootcamp.uxdesign.cc/8-skills-you-must-have-as-a-front-end-developer-in-2022-6b851e05d990

Hipages 2023, 2023. *Hipages.* [Online]   
Available at: https://hipages.com.au/tradie/success-guide/top-tips-for-quoting/#:~:text=On%20average%2C%20tradies%20are%20spending,full%20day's%20work%20every%20week.  
[Accessed 01 2023].

Karp, P., 2019. *The Guardian.* [Online]   
Available at: https://www.theguardian.com/australia-news/2019/dec/03/australian-students-maths-performance-falls-to-oecd-average-in-worst-result-since-2000  
[Accessed 01 2023].

Neurosys, n.d. *Neurosys.* [Online]   
Available at: https://neurosys.com/blog/why-people-say-unity-engine-is-bad#:~:text=Unity%20is%20a%20multi%2Dplatform,without%20extensive%20programming%20knowledge%20needed.  
[Accessed 01 2023].

Unity, 2023. *Unity.* [Online]   
Available at: https://unity.com/  
[Accessed 01 2023].

Unity, n.d. *Unity Documentation.* [Online]   
Available at: https://docs.unity3d.com/Manual/UI-system-compare.html  
[Accessed 2023].