In today's world, many of the objects people use on a daily basis are not only electronic, but controlled on some level by a computer. Phones, kitchen appliances, cars and digital applications are now essential to life and are powered by massive code bases that require constant maintenance and updating to stay useful, functional and secure. Releasing broken code or failing to make meaningful updates and additions to products in a reasonable time period can spell the end for companies. But changes to the code can be an incredibly difficult task for developers, especially when applications like Microsoft Windows can have upwards of 40 million lines of code. How can a developer who just joined the team learn how such a massive application works, especially to the level that they can then modify and add to the code without breaking everything? It's already hard enough for developers who have been on the team for months or years. Right now, the answer is through comments in the code and pages upon pages of documentation. But comments can be unreliable, and documentation hard to read and understand. Developers need a tool for easily understanding huge, hard to comprehend code bases. And right now, there isn't one. Our product, Codenser, will be able to fill this void.

As a useful metaphor, imagine a car mechanic who has been in the workforce for a couple years. They've studied some sort of engineering in school, they've worked with a number of different cars fixing and possibly modifying them; maybe they've even built their own car from scratch, engine and all. It is safe to say that this mechanic has a pretty in-depth knowledge of how engines in general work, basic electronics and hydraulics, all the things that go into building a car. Now that mechanic decides to switch gears and joins NASA, working on a team that is building a brand-new spaceship. To help get them up to speed, the team provides them with a complete schematic of the rocket, complete with a 10000-page booklet of documentation explaining everything. This is kind of like what it is to be a programmer joining a new team. All developers have a basic understanding of how code works, how computers work, how applications work, etc. But this does not mean that they can jump into a massive new project and immediately understand how everything works and make changes to it. Even the best rocket scientist in the world can't just look at a new rocket and immediately start improving or fixing it. But what if instead of reading a ridiculous amount of mind-numbing documentation, the mechanic could press a button on the rocket and see a diagram light up highlighting exactly which components are being activated, how they work together, and get a summary description of what each part does. That would make familiarizing themselves with how everything works on the new project so much easier.

This is what Codenser aims to do with code. A new developer can run the code base through the application, which will return to them pseudocode, execution traces for all functionality, and organize it all into an intuitive, interactive diagram. The developer could then click on the various functionalities of their project and see exactly which classes are being used, which functions are running and in what order, and get pseudocode descriptions of what these functions are doing. There will be technical challenges of course. For example, writing good pseudocode can already be difficult, so how can that be automated so that a computer can read code and deduce what is important and what it is trying to do? And can it then write that in a meaningful way, maybe in a couple English sentences? Execution traces can be long and hard to follow, how can that be portrayed or summarized in a meaningful way? And how can an

application with millions of lines, hundreds of thousands of functions, and thousands of classes be visualized in a useful way? These are all obvious hurdles, but we as a team believe they can be overcome and Codenser will become an application used by development teams everywhere.