5 Case Study: The Database Guru

Rajiv worked for five years as a software engineer at Felicity Software, a company that sells database systems to consumers and small businesses. During his time at Felicity, Rajiv implemented some clever software optimizations that significantly improved the performance of the database systems sold by Felicity. The company treated the optimizations as trade secrets. It labeled the software as confidential information and took measures to restrict access to the source code to Rajiv and a few other software engineers who helped Rajiv implement and test the system.

On his first day with Felicity, Rajiv had signed an employee confidentiality and proprietary rights agreement. Signing this agreement was a condition of his employment. In this agreement Rajiv had indicated that he understood and acknowledged the following:

1. He would have access to confidential, secret, and proprietary information related to Felicity's business.
2. Felicity placed "great competitive importance and commercial value" on its ability to reserve confidential information for its exclusive use.
3. Confidential information he developed would be covered by this agreement.
4. He would not "directly or indirectly disclose, publish, communicate or make available" confidential information to anyone outside of the company.
5. His obligations under the agreement would "continue during and after his employment" by Felicity [38].

After completing the database project at Felicity, Rajiv began looking for the opportunity to join a software start-up. With a good reputation in the tight-knit local tech community and strong references from coworkers and managers who praised his talent, work ethic, honesty, and teamwork skills, Rajiv quickly found employment at Unrelated.com, a start-up company in a nearby office park.

Unrelated.com plans to support people doing genealogical research. It is developing a proprietary database system that will be used to store genealogical information. Unrelated.com has no plans to sell the database software. Instead, its income will be derived from the monthly fee users pay in order to access the company's database through the Web or a mobile app. The sales and marketing team estimates that within three years Unrelated.com could reach a steady state of 50,000 subscribers paying $10 a month, as long as the system performs well. Up to 20 percent of the customers may be using the system during peak periods, which means in order to reach a steady state of 50,000 customers, the database system must be capable of quickly responding to queries from 10,000 users accessing the system simultaneously. If the performance of the system falls short of this goal, fewer people will subscribe to the service. For example, a system that can provide good service to only 5,000 concurrent users will support only 25,000 subscriptions.

Rajiv's title at Unrelated.com is vice president of software, and he earns a salary of $150,000 supervising several teams of software developers. One of these teams is responsible for implementing the genealogical database system and optimizing its performance. As the project progresses, Rajiv realizes that some of the technical solutions proposed by the members of his team are not as good as the solutions he developed for Felicity's database product. The team's preliminary software implementation is unlikely to result in a database system with satisfactory performance when it is being accessed by more than about 5,000 users. In Rajiv's judgment significant optimizations are needed before the system is ready to deploy if Unrelated.com is to reach its goal of 50,000 monthly subscribers.

Rajiv believes he has two viable alternatives. The first option is to follow a "clean-room" strategy. He would isolate himself from the team's work product, but he would provide the team with publicly available information on database optimization strategies—in other words, the books, journal articles, and conference papers he benefited from when he implemented the database system optimizations at Felicity. Based on his expertise, Rajiv would also provide the team with realistic performance targets that the various components of the database system should be able to meet.

The second option is to become personally involved in making the necessary performance improvements. Rajiv believes that if he asks the right questions, in fairly short order he can get the team to rediscover the optimizations he developed for Felicity's database product without actually telling them how to do it.

Which option should Rajiv take?