No. 18-956

In The

Supreme Court of the United States

GOOGLE LLC,

Petitioner,

v.

ORACLE AMERICA, INC.,

Respondent.

On Writ Of Certiorari To The United States Court Of Appeals For The Federal Circuit

BRIEF OF SOFTWARE AND SYSTEM DEVELOPERS AND ENGINEERS FOR UNITED STATES GOVERNMENT AGENCIES AS *AMICI CURIAE* SUPPORTING GOOGLE AND URGING REVERSAL

ANDREW P. BRIDGES
FENWICK & WEST LLP
801 California Street
Silicon Valley Center
Mountain View, CA 94041
(415) 875-2389
abridges@fenwick.com

Counsel for the Amici Curiae

TABLE OF CONTENTS

TABLE OF AUTHORITIES

INTEREST OF THE *AMICI CURIAE*[1]

The *amici curiae* who present this brief are experienced computer science and software development professionals who have built and maintained complex computing systems, for government agencies and the private sector, relying upon fundamental principles of open interoperability of software through application programming interfaces. They have all worked in the United States Digital Service and have experience working with very large systems that help government operate more efficiently and provide better service to citizens. They have a stake in the consistent and correct determination of the scope of copyright protection that applies to interfaces of computer programs, including the Java interfaces at stake in this case.

Each of the *amici* relies on the availability of open interfaces in developing and by adapting legacy systems to serve the government's, and the public's, needs. Managing, maintaining, and updating systems typically requires the introduction of new components that are compatible with or interoperate with pre-existing computer products, platforms and services. Interoperability is the very foundation of the Internet, the Web, and of countless devices and services that depend upon

[1] No counsel for any party authored this brief in whole or part, and no person other than the *amici curiae* or their counsel made a monetary contribution to the preparation of this brief. Both Petitioner and Respondent have consented in writing to the filing of this brief.

them. It is also key to the longevity and adaptability of very large computer systems.

The *amici* believe that the Federal Circuit's decisions disturb well-established principles upon which they and many other developers, and the agencies and other employers they have worked for, have built systems for over two decades.

The *amici* believe that computer program code deserves copyright protection. But they see the Federal Circuit's decisions as posing a serious threat to future innovation and competition in information and communication technology and service sectors. Those sectors have thrived until now because of the smooth interoperation of systems that rely upon many components from different sources and upon competition among sources of interchangeable and interoperable components. The Federal Circuit's decisions threaten long reliance of developers upon the free and unhindered availability of programming interfaces in the construction of complex systems.

These *amici* cannot stress enough that affirmance of the Federal Circuit's decisions would create calamity among software developers, systems integrators, vendors, and users of virtually all kinds of multi-component systems that rely upon software to operate.

The *amici* are all individuals. They file this brief in their own names and to express their personal views. None of their views or statements in this brief should be attributed to any entity with which they are associated now or they have been associated with in the past.

The *amici* are the following:

- Alex Gaynor served as a software engineer with the United States Digital Service from 2015 to 2017, principally at the Departments of Veterans Affairs and State, where he worked on improving large government computing systems to process veterans' disability benefit claims more quickly and to help veterans apply for health insurance online. After working for the United States Digital Service, he was an engineer working on security of the Firefox browser. He is currently the Chief Information Security Officer at Alloy. He has previously served as a member of the board of directors of the Python Software Foundation and Django Software Foundation.

- Eric Benson served as a software consultant with the United States Digital Service in 2015, principally at the Social Security Administration. He was previously a software architect at Amazon.com and Lucid, Inc., and was the principal architect of Lucid Common Lisp. He has over thirty years' experience in software development, including programming language implementation, software portability, and Internet commerce.

- Liyan David Chang was a software engineer on the Healthcare.gov rescue team and replaced the front-end and login systems with API-compatible successors. He then served two years with the United States Digital Service, partnering with a number of agencies including the Internal Revenue Service, the Department of Justice, and the Department of Homeland Security, and he was briefly Acting Director of Engineering at USDS. Currently, he is a Senior Software Engineer at Devoted Health, a Medicare Advantage health insurance firm. He was the co-founder of a Y Combinator- and Andreessen Horowitz-funded startup and holds a degree in Computer Science from Massachusetts Institute of Technology.

- Shauni Deshmukh served as a software engineer with the United States Digital Service from 2015 to 2016. She is currently Chief Technology Officer at Tettra. She has over 15 years of software engineering experience, including previous roles at Twitter and MITRE. She holds a B.S. in Computer Science from Massachusetts Institute of Technology.

- Scott Haselton was a member of the United States Digital Service from 2016 to 2019 as a software engineer. He was the lead for major modernization efforts at the Centers for Medicare and Medicaid Services that was transitioning the yearly $500 billion fee-for-service payments towards value-based payments. He has previously worked at various startups in finance and entertainment as an engineering manager and core contributor. He has been developing software professionally for the past twenty years.

- Shaun Verch is a site reliability engineer with ten years of experience in operations, database systems design, and distributed applications. He served with the United States Digital Service from 2016 to 2018, aiding in the successful launches of qpp.cms.gov and login.gov. He has also worked at multiple startups on the leading edge of technology, including MongoDB, which has now gone public, and PlanetScale, where he currently works.

- David Koh served as a software engineer with the United States Digital Service from 2016 to 2019, primarily at the Department of Homeland Security and Department of Health and Human Services. He is currently a founder and the Chief Technology Officer at Slipstitch, a civic software company. David has over ten years of experience as a software engineer and leader of software engineering teams, including serving as the Director of Engineering at OkCupid.

- Julie Meloni served as a product manager with the United States Digital Service from 2016 to 2018, first at the Department of Veterans Affairs and later as Director of Product Management and Strategy/Operations for USDS as a whole, where she primarily helped orchestrate cross-agency, cross-functional teams to address emerging technical concerns. She has worked in the software industry since 1994 as an engineer, engineering manager, and product manager, and is currently Head of Engineering and Applied

Science in the U.S. Defense division of Improbable.

- Ellen Ratajak came out of retirement to join the United States Digital Service in 2015. For seven months, she was part of the team supporting the Department of Veterans Affairs. Before that, she was one of the first engineers at Amazon.com, forming its fulfillment center software team and leading it through some of its most crucial growth phases (1996-2000).

- Shelby Switzer is a software engineer with the United States Digital Service at the Department of Health and Human Services. She has led API and integration teams in the healthcare tech and Internet of Things industries. She has worked with Code for America brigades and other civic and community tech organizations in multiple cities across the United States, and she co-organizes REST Fest, an international conference series on web architecture. Over the past seven years she has spoken regularly at technology conferences across three continents and written for various publications about API strategy, civic technology, and open source software.

- Victor Garcia is currently a Senior Director of Software Engineering at The Walt Disney Company; serves as Vice Chair of the Board of Directors of the MATHCOUNTS Foundation; and is a co-founder and advisor of &Partners: an ethical technology solutions firm and a service-disabled veteran-owned small business certified under section 8(a) of the Small

Business Act, 15 U.S.C. § 637(a). Previously, he served in the United States Digital Service at the White House as an Engineering Director for the U.S. Citizenship and Immigration Services, and worked on projects across the Departments of Homeland Security, Veterans Affairs, Health and Human Services, and the Railroad Retirement Board. Before working at USDS, Victor was a software engineering lead at Yahoo and worked on large scalable web systems that powered some of the web's most visited sites. He holds a degree in Computer Systems Engineering from Boston University.

- Andy Brody served as a software engineer with the United States Digital Service from 2016 to 2020, largely at the Department of Homeland Security and at the General Services Administration, where he led the infrastructure team behind Login.gov. He previously worked on cloud infrastructure at Stripe, where he helped the company handle exponential growth and also started Stripe's information security team. He has delivered talks at several prominent technology conferences, and he is the inventor of a patented software process in computer security. He holds a bachelor's degree in computer science from Harvard University.
- Aaron Wieczorek served at the United States Digital Service from 2017-2019 in roles such as software engineering and as the General Counsel, working primarily at the Department of Veterans Affairs, the Department of

Health and Human Services, and the Office of Management and Budget. He is currently a software engineer at Rebellion Defense. He has more than ten years of experience as both a software engineer and lawyer working at the intersection of technology and law in both the public and private sectors and has spoken at many technology conferences and served on non-profit boards.

- Marianne Bellotti is an author and expert on modernizing legacy computer systems. She served with United States Digital Service from 2015 to 2019 and the United Nations from 2013 to 2015. She is currently studying the behavior of fake accounts on social networks.

- Shane Russell served as a software engineer with the United States Digital Service from 2016 to 2019, at the Department of Veterans Affairs. He is currently a Senior Software Engineer at One Medical. He has over ten years' experience as a software engineer, having delivered several talks at software engineering conferences.

  Jordan Kasper is a Digital Service Expert at the United States Digital Service and has been working within the U.S. Department of Defense since 2017. He works on systems such as the Joint Enterprise Defense Infrastructure (JEDI) Cloud, the U.S. Army Cyber School curriculum, and the Global Combat Support System–Marine Corps (GCSS-MC). Before joining the United States Digital

Service, Jordan worked as a software engineering instructor, a Developer Advocate with IBM, a Systems Analyst at the University of Texas System, and as a software engineer at various technology startups.

- Andrew Nacin served as a software engineer with the United States Digital Service from 2015 to 2019. He worked on projects at more than a dozen government agencies, principally on the immigration and refugee programs, and served as Senior Advisor to the Administrator of the United States Digital Service from 2017 to 2019. He has served as Lead Developer of the WordPress open source software project, which powers more than a third of all websites.

- Harlan Lieberman-Berg has been a software engineer with the United States Digital Service since 2017, posted at the Department of Defense. He has previously worked as an information security architect for Akamai Technologies, as well as in engineering management roles at various ad-tech startups. He works in a variety of roles in the free software and open source communities, including as a Debian Developer, and has previously served as the Gentoo Linux kernel security lead. He has over a decade of experience in software development and focuses on information security and engineering ethics.

- Elliott Wilkes served as a product manager for the United States Digital Service from 2016-2019 primarily working at the Departments of State and Defense. He is currently at the Defense Digital Service as a digital service liaison working with the United Kingdom Ministry of Defence and helped launch its UK Defence Digital Service. He has over ten years of experience working as a product manager and technologist, advising and consulting for the United Nations, Forsa, Mercy Corps, and other international organizations in the Middle East, Africa, Europe, and the U.S.

- Lucas Merrill Brown served as a software engineer with the United States Digital Service from 2016 until 2018. He worked with the Centers for Medicare and Medicaid Services as the Chief Technology Officer of the Quality Payment Program. He is currently the lead software engineer at Myst AI, a startup using machine learning to increase renewable energy adoption and reduce carbon emissions. He has eight years of experience as a data scientist and engineer working on social impact causes. He received a D.Phil. degree (a Ph.D. equivalent) from Oxford University researching statistical models of how consumers adopt new clean energy technologies.

- Judy Siegel is currently Director of Digital Design and User Experience at Dow Jones, and is based in New York City. Previously, she was a User Experience Designer with the United States Digital Service from 2017 to 2018, and worked at both the Centers for

Medicare & Medicaid Services and the Department of Veterans Affairs.

SUMMARY OF ARGUMENT

Computer programs, and complex systems that rely on them, achieve compatibility and interoperability with each other through a multitude of specifically defined interfaces. The use of computer program interfaces of others for compatibility and interoperability purposes is both ubiquitous and essential to the operation of information and communication technologies and infrastructures such as the systems we have worked on for the government and the private sector. This fact has become even more so in today's ever more highly networked world. The freedom to utilize, implement, re-implement, and extend existing interfaces, without fear of a veto based on the copyright statutory monopoly, has been the key to competition and progress in the computer, information technology, communication technology, and networking fields since their beginning. The Federal Circuit's decisions below call into question a bedrock legal foundation that has caused innovation to flourish in those fields and in our jobs.

The *amici curiae* are deeply concerned that the Federal Circuit's decisions hand some copyright holders a power that the Copyright Act did not provide, and that Congress did not envision: the ability of the copyright holder to monopolize systems, processes, and methods of operation of others merely because those systems, processes, and methods interact with or otherwise utilize an interface embodied in the copyright holder's product. The result of the Federal Circuit's unwarranted expansion of copyright law, contrary to express statutory limitation of the Copyright Act, and violating the boundary between copyright and patent, will be that technology and communications infrastructures, systems, and services will become more fragmented, less standardized, and less interoperable, all to the detriment of technical progress and efficiency, and of the Progress of Science and useful Arts. *See* U.S. Const. Art. I, Sec. 8, Cl. 8.

The Federal Circuit's decisions disrupt a settled expectation in the information and communications technology communities that programming interfaces are uncopyrightable functional elements. This disruption has introduced new uncertainty that threatens chaos in the day-to-day work of those who build, operate, and maintain important and complex systems that support all sectors of the society and economy.

These *amici*, who spend their days, and have spent years of their professional careers, designing complex and critical systems for a wide variety of public and private uses recognize the urgency and importance of this Court's correct application of copyright law in service of copyright's Constitutional purpose.

These *amici* leave to others the articulation of the principal legal arguments. These *amici* focus on explaining the reality of the systems development world so that the Court may understand what is at stake. They urge the Court to reverse the decisions of the Court of Appeals for the Federal Circuit.

ARGUMENT

I. THE USE OF FUNCTIONAL PROGRAMMING INTERFACES LIKE THOSE AT ISSUE IN THIS CASE IS UBIQUITOUS BECAUSE IT IS NECESSARY TO MODERN SYSTEM DESIGN.

Interoperability is the foundation of an extraordinary range of products, systems, and services. The Internet, telephone systems, national defense networks, emergency response systems, and—closer to this case—a personal computer and a mobile telephone all rely upon contributions and components from numerous sources.

Those, like us, who design, maintain, and update complex systems depend upon interoperability of components of those systems. The expectation of a lawful freedom to design products to work with others, by relying upon the other products' functional declarations, is the bedrock on which we build our own creative implementations.

As professionals who design, maintain, and update complex systems, we work hard to identify the best combinations of components, functions, and environments to produce the best effects. We combine different components, from different sources, to bring out the best potential that each component has both individually and *in relation to other components* of the systems that we assemble. When we design systems by bringing many components together, we can produce benefits that producers of the individual components may not have intended or even envisioned: the system is far greater than the sum of its components. Interoperability is thus an especially valuable engine of innovation.

This ability to combine components freely also promotes competition, which in turn drives progress and innovation. Interoperation of components from many different sources allows developers to identify the best products for different functions or roles within a system, without forcing the developers to choose among only a few rival fully integrated systems. We need to have as many tools as possible. Interoperability avoids lock-in effects that would result from incompatibility among products of different sources.

Moreover, many complex systems have long lifetimes, especially in government where many of us served. They require significant work to keep them relevant to current needs as the demands on them grow. They must accommodate more data, must communicate more quickly, must carry out new functions, must adapt to new threats, and must otherwise struggle against other challenges of age. Those who operate and maintain massive systems must constantly update them by replacing old components and introducing new ones. Sometimes components become obsolete, perhaps because the technology is too old, their provider failed to support the product, or a new product works more efficiently. In that event one must swap in a new component to replace the old one in a system environment. The new component must work with other parts of the system in the same way as the old component did. Otherwise a massive reprogramming effort of all the surrounding components would be necessary for them to work properly with the new one. The inability to rely upon consistent functional programming interfaces in replacing a component would impair operation of the system, cause delays, be costly, and create substantial new risks of error in the integration. For those of us who maintain giant systems with many components, having to rebuild an entire system when replacing a component would be a frightening prospect. The undertaking would be extraordinarily costly and inefficient, which could lead to abandonment of previous investment or continuing operation of vulnerable and poorly functioning platforms.

In addition, when developing new products, one needs to optimize the development by testing them in simulations before committing them to real-world operations. One needs to use programming interfaces of other interoperable products in a system to design appropriate tests, to ensure that the products communicate appropriately without real-world risk.

One prominent example demonstrates how important reimplementing programming interfaces can be to resolving problems with government technology while minimizing interruptions—"downtime"—for the public. When the federal government launched healthcare.gov, it was a complex software system, built from many components. One component, which was responsible for authenticating users, caused over half of the notorious downtime of the system in the first year. The component was a commercial-off-the-shelf authentication system from Oracle. Healthcare.gov had used it in a way that it was not suited for, leading to the poor performance and downtime. To rectify this, a team built a replacement authentication system known as the Scalable Login System (SLS).

To minimize the risk of deploying this change, the engineers building SLS needed to have it implement the same programming interfaces as the previous authentication system from Oracle. This meant that no other component of healthcare.gov needed to change with the deployment of SLS, which in turn gave the healthcare.gov team increased confidence that this change was low-risk and that the team could quickly proceed with change.

II. THE FUTURE PROGRESS OF SCIENCE AND USEFUL ARTS, THROUGH THE CONTINUATION OF DECADES OF BREATHTAKING INNOVATIONS, DEPENDS UPON A DETERMINATION THAT PROGRAMMING INTERFACES LIKE THOSE HERE ARE FUNCTIONAL AND OUTSIDE THE STATUTORY MONOPOLY OF COPYRIGHT.

Referring to our example of the healthcare.gov system, had the conventional understanding and expectation then been different, namely that programming interfaces were subject to copyright, the developers would have needed to change many other components of the healthcare.gov system. The process would have been like replacing one leg of a chair with a shorter one, causing a need to shorten all the other legs to make the chair stable and balanced again.

But for the team's ability to rely upon consistent functional programming interfaces, it could not have fixed the problem. This would have prolonged healthcare.gov's struggles at the expense of citizens trying to use the service and at cost to the taxpayer.

If the provider of the component being replaced could exercise a veto power over the substitution of the component by a replacement, that provider could leverage its copyright monopoly to exact a royalty over other, interoperating, components in the environment merely because those components had functioned together in a system. Demanding a royalty to authorize the functioning of other products is more akin to exploiting a patent; it is not appropriate for a copyright owner to exercise that power.

Both replacing an old component and adding a novel component require that the new components fit seamlessly into the pre-existing system environment. That means the new and the old must operate together as reliably as before.

Google's use of the functional programming interfaces at issue in this case has been typical of uses by virtually all software developers and systems designers, including us.

The key here is function, not expression. The programming interfaces are important as tools by which we can connect software and other components so that they can communicate properly and work with each other. It doesn't matter to us as developers how those declarations are expressed; what matters is that they are consistent. (It is useful for expressions to reveal their function, but it is the function that counts.) For developers, relying on a declaration to identify a component for interoperation is like relying on a book title to refer to a book, enabling someone to identify the book to locate it and read it. While a book, like an entire software program, may be subject to copyright, a title of a book has long been understood as not subject to copyright. *See* 37 C.F.R. § 202.1 (words and short phrases such as names, titles, and slogans are not subject to copyright).

Until the Federal Circuit's decisions against Google, we, like countless others, believed the reimplementation of declarations of others was permissible. We understood copyright law to contain a bright-line distinction between original creative expression on the one hand and functional elements of a work on the other hand, which both section 102(b) of the Copyright Act, 17 U.S.C. § 102(b), and the related merger doctrine establish. We thought that the reimplementation of declarations in new products to create interoperable or compatible systems was accepted and uncontroversial, allowing us to focus purely on engineering decisions. Reversal of the Federal Circuit's decisions is necessary to allow us the clear guidance resting on section 102(b) that we need to keep doing our jobs and that future government technologists will need to do their jobs.

III. WHILE THE FAIR USE DOCTRINE SUPPORTS THE LAWFUL REIMPLEMENTATION OF DECLARATIONS, IT IS NOT ENOUGH TO ASSURE THE SAFETY THAT DEVELOPERS AND ENGINEERS NEED IN CREATING, MAINTAINING, AND UPDATING LARGE AND VITALLY IMPORTANT SYSTEMS.

Application of section 102(b) and the merger doctrine spares us the need to guess how courts would react and to consult lawyers for legal opinions about risky balances of fair use factors and interests.

As this Court has recognized, the fair use doctrine typically applies on a case-by-case basis. "The task is not to be simplified with bright-line rules, for the statute, like the doctrine it recognizes, calls for case-by-case analysis." *Campbell v. Acuff-Rose Music, Inc.*, 510 U.S. 569, 577 (1994) (citing *Harper & Row Publishers, Inc. v. Nation Enterprises*, 471 U.S. 539, 560 (1985)). The fair use statute and doctrine do not provide a "rigid, bright-line approach." *Campbell*, 510 U.S. at 584.

The fair use doctrine is not enough to protect and foster the innovation we have seen over decades. A bright-line approach is essential to the countless decisions that must occur every day, in technology development across all industries and endeavors, to build, maintain, or update systems. It is not practical to subject those decisions to fair use analysis and debate. It is not safe for a developer or systems integrator to rely solely on the fair use doctrine as a basis for investing countless hours in building an innovative and creative product, and especially for a company to assemble or update a complex system with numerous components that must work together. The risk of facing an injunction would be intolerable. The potential of dominant component suppliers to use a copyright statutory veto power to destroy a complex integration project, or to demand tribute, because a court may unpredictably balance interests, *see id.*, is unacceptable. It is not possible for system developers or engineers to "clear" copyright permissions to use the functional interfaces of countless components and to innovate the way they have innovated for decades. And given the solely functional reason for using those programming interfaces, there is no reason to.

CONCLUSION

For the reasons we have explained above, these *amici curiae* urge the court to reverse the decisions of the Federal Circuit and to rule that Section 102(b) of the Copyright Act precludes a copyright statutory monopoly in the functional programming interfaces, the declarations, that allow interoperability and compatibility of software and other components of systems.

Respectfully submitted,

ANDREW P. BRIDGES
FENWICK & WEST LLP
801 California Street
Silicon Valley Center
Mountain View, CA 94041
(415) 875-2389
abridges@fenwick.com

Counsel for the Amici Curiae